Inexpensive Backup Solutions for Open Systems

Covers Windows, Linux, UNIX, and OS X



Backup & Recovery

O'REILLY®

W. Curtis Preston

Backup & Recovery

W. Curtis Preston

Editor

Mike Loukides

Copyright © 2009 O'Reilly Media, Inc.



O'Reilly Media

Preface

I hope you learn half as much reading this book as I did writing it. This was quite an interesting project, where we took the original book and expanded its scope so much that we had to change its title. I wrote *Unix Backup and Recovery* seven years ago, and a lot has changed since then—both in the industry and in my life. The biggest change in the industry has been the proliferation of Windows, Mac OS, Exchange, and SQL Server in the data center. (I never saw the Apple Xserve coming.)

The biggest change for me has been having my eyes opened to backup and recovery applications beyond those considered "traditional." It's true that I spend most of my professional life consulting with large companies that spend enough on backup software and hardware to fund a small army. I enjoy doing that. It's very rewarding to show a company how to save millions of dollars a year and make their backups and restores faster and more reliable in the process. (By the way, if you need help with your backup system, drop an email to curtis@backupcentral.com—that's what I do for a living.)

I also spend a good deal of the time traveling the world speaking to users about how to do this themselves. And when I do, I always get questions like these:

I got a quote for backup software from XYX, and they want $XXXX for backup software! Where am I supposed to get that kind of money!?

I couldn't afford backup software from XYZ, so we bought ABC instead, and it stinks. Can you recommend something better?

None of the commercial utilities can back up my MySQL or PostgreSQL database. How do I do that?

How do I do bare-metal recovery on ABC operating system?

Aren't there open-source utilities that do this kind of thing?

So while I'm actually preparing to write my next book on how to select, install, and manage commercial backup software systems, I felt that this book needed to come first. This book is aimed at the people who feel that the commercial software products aren't meeting all their needs.

Perhaps you're a small shop that can't spend $10,000 just to get decent backup software. Perhaps you're already using a commercial backup software package, but you don't want to spend thousands of dollars on their agent to back up your DB2 databases, or you can't find anybody to back up your MySQL or PostgreSQL databases. This book is about giving you options—*free* options.

Almost everything I talk about in this book is either included with your operating system or application, or is available as an open-source project. (The commercial products I do mention cost only $99.) You may be amazed at what you can do for free or almost free.

I Wish I'd Had This Book

I wanted to write a book that would ensure that no one would ever have to start from scratch again, and I believe that my contributors and I have done just that. It contains every backup tool that I wish I had when I first entered the backup business and every lesson and trick that I've learned along the way. It covers how to back up and recover everything from a basic Linux, Windows, or Mac OS workstation to a complicated DB2, Oracle, or Sybase

database—and a lot of things in between. Whether your budget barely stretches to cover the cost of the backup media or allows you to buy a silo bigger than your house, this book has something for you. Whether your task is to figure out how to back up, with no commercial utilities, an environment such as the one I first encountered or to choose from among more than 50 commercial backup utilities, this book will tell you how to do it. With that in mind, let me mention a few things about this book that are unique.

Only the Recovery Matters

As my friend Joe Fitzpatrick used to tell me, “No one cares if you can back up—only if you can recover.” Yet how many backup chapters have you read that dedicate less than 10 percent to recovery? You won’t find that in this book. I have tried very hard to ensure that recovery is given equal treatment.

Products Change

Some people may be surprised that there are no product names mentioned in the commercial backup section. I did this for several reasons, the main one being that products change constantly. It would be impossible to keep this book up to date with more than 50 backup products that are available for Unix alone. In fact, the book would be out of date by the time it hit the shelves. Instead, this book explains the *concepts* of commercial backup and recovery software, allowing you to apply those concepts to the claims that the vendors are currently making. Up-to-date information about specific products is available on http://www.backupcentral.com.

Backing Up Databases Is Not That Hard

If you're a database administrator (DBA), you may not be familiar with the commands necessary to back up your database. If you're a system administrator (SA), you may not be familiar with the architecture of the database platform your DBA is using. Both concepts are explained in detail in this book. I explain the backup utilities in plain language so that any DBA can understand them, and I explain database architecture in such a way that an SA, even one who has never before seen a database, can understand it.

Bare-Metal Recovery Is Not That Hard

One of these days you will lose the operating system disk for an important system, and you will need to recover it. This is called a *bare-metal recovery*. The standard recovery method described in many backup products' documentation is to install a minimal operating system and restore on top of it. This is the worst possible method to do a bare-metal recovery of a system; among other problems, you end up overwriting some of the system files while the system is running from the very disk to which you are trying to restore. The best ways to do bare-metal recoveries for AIX, Solaris, HP-UX, Windows, Linux, and Mac OS are covered in detail in this book.

How This Book Is Organized

This book is divided into six parts, which are described in the following sections.

Part I

Part I of this book contains just enough information to whet your backup and recovery appetite.

Chapter 1, The Philosophy of Backup

> Describes my philosophy about backup, such as why you should back up, and a little bit about how to do it, too.

Chapter 2, Backing It All Up

> Goes into detail about the essential elements of a good backup and recovery system.

Part II

This section covers the basic backup utilities that are available to back up your system, and several open-source backup systems to help you manage those backups.

Chapter 3, Basic Backup and Recovery Utilities

> Covers the basic backup and recovery utilities you're likely to find in Unix, Windows, or Mac OS, including `dump`, `tar`, `cpio`, `dd`, `ditto`, `ntbackup`, and `rsync`.

Chapter 4, Amanda

> Covers the ever-popular Advanced Maryland Disk Archiver, or Amanda.

Chapter 5, BackupPC

> Explains the disk-only backup system called BackupPC, which can actually back up far more than just your PC.

Chapter 6, Bacula

> Covers Bacula. It roams the data center at night and sucks the vital essence from your computers.

Chapter 7, Open-Source Near-CDP

> Covers three near continuous data protection (near-CDP) products, including `rsync` with snapshots, `rsnapshot`, and `rdiff-backup`.

Part III

If you have outgrown the capabilities of free utilities or would just like to take advantage of new backup and recovery technologies, you'll need to look at a commercial product. You should also know about the latest hardware that is on the market to assess your full range of backup and recovery options.

Chapter 8, Commercial Backup Utilities

> Is your guide to the hundreds of features available in the over 50 commercial backup products available on the market today, allowing you to make an educated purchase decision.

Chapter 9, Backup Hardware

> Explains the many different types of backup hardware available today, and provides criteria to help you decide which type of backup drive is right for you.

Part IV

A bare-metal recovery is the fastest way to bring a dead system back to life, even if its operating system drive is completely destroyed.

Chapter 10, Solaris Bare-Metal Recovery

Explains Sun's flash archive product, which is the Solaris equivalent of AIX's `mksysb`.

Chapter 11, Linux and Windows

Explains a number of procedures and tools that can be used to perform bare-metal recovery of both Linux and Windows systems. It includes a discussion of Ghost for Linux (G4L) an open-source ghosting product.

Chapter 12, HP-UX Bare-Metal Recovery

Covers the `make_net_recovery` and `make_tape_recovery` tools, which now come with HP-UX to perform bare-metal recoveries.

Chapter 13, AIX Bare-Metal Recovery

Discusses AIX's `mksysb`, probably one of the oldest and best-known bare-metal recovery tools.

Chapter 14, Mac OS X Bare-Metal Recovery

Covers how to perform your own bare-metal recovery of a Mac OS X machine.

Part V

This section explains in plain language an area that presents some of the greatest backup and recovery challenges that a system administrator or database administrator will face—backing up and recovering databases.

Chapter 15, Backing Up Databases

> Explains database architecture while relating each architectural element to the appropriate term in DB2, Exchange, Informix, MySQL, Oracle, PostgreSQL, SQL Server, and Sybase. This chapter will be your friend if you're an SA who's afraid of databases or a DBA learning a new database.

Chapter 16, Oracle Backup and Recovery

> Explains how to perform Oracle hot backups using `rman` or user-managed backup.

Chapter 17, Sybase Backup and Recovery

> Shows how to use the backup server to back up Sybase ASE.

Chapter 18, IBM DB2 Backup and Recovery

> Explains how to back up and recover DB2 databases.

Chapter 19, SQL Server

> Explains how to back up and recover SQL Server databases.

Chapter 20, Exchange

> Explains how to back up and recover Exchange databases using the built-in `ntbackup` plug-in for Exchange.

Chapter 21, PostgreSQL

Explains how to back up and recover PostgreSQL databases.

Chapter 22, MySQL

Provides an overview of the various backup and recovery options available for MySQL.

Part VI

The information contained in this part of the book is by no means unimportant; it simply wouldn't fit anywhere else!

Chapter 23, VMware and Miscellanea

> Includes VMware backups, the oft-debated "live filesystem dumps" question, and even some backup poetry.

Chapter 24, It's All About Data Protection

> Provides some food for thought, discussing the fact that backups are not the answer to all problems; you should also be thinking about other areas of data protection, such as archiving, disaster recovery, and storage security.

What's New in This Book

See preceding section. Seriously, this book has about 75 percent new material when compared with *Unix Backup & Recovery*. Some chapters in the first book were completely rewritten for this book. Here are the highlights of those changes:

A new philosophy

: This book reflects my new backup philosophy, which is that it's all about disk—especially for smaller shops.

New backup commands

: We've added `ntbackup`, `ditto`, and `rsync` to the basic utilities chapter.

Amanda

: The Amanda chapter is completely updated to reflect the developments of the past seven years.

Commercial utilities

: The commercial utilities chapter has been updated to reflect the advances in backup and recovery in the past seven years.

HP-UX

: The `make_net_recovery` and `make_tape_recovery` tools have changed, and so has the chapter covering them.

Backup hardware

: Boy, has hardware changed in seven years! Disk targets, virtual tape libraries, and data de-duplication systems. I cover it all.

There are eleven *completely new chapters*, significantly expanding the scope of this book. Here are the topics covered in these new chapters:

DB2

: How to back up DB2 using its built-in capabilities

Exchange

: How to back up Exchange using `ntbackup`

SQL Server

: How to back up SQL Server using its built-in capabilities

MySQL

: How to back up and recover MySQL databases based on the MyISAM, InnoDB, and NDB storage engines

PostgreSQL

: How to back up and recover this popular open-source database using either `pg_dump` or `pg_dumpall`

BackupPC

: How to use BackupPC, a completely disk-based backup and recovery system with a web frontend

Bacula

: How to use Bacula, an open-source backup product that roams the datacenter at night and sucks the vital essence from your computers

Near-CDP

: How to use snapshots and replication to make backups

Solaris

: How to do bare-metal recovery using flash archive

Linux and Windows bare-metal recovery

- How to use a Linux LiveCD or Ghost for Linux to perform bare-metal recovery of Windows and Linux operating systems

Mac OS X

- How to use the built-in, bare-metal recovery in OS (it isn’t too hard)

What's Missing?

For various reasons, some chapters from *Unix Backup & Recovery* did not make it into this book. All of the following chapters are now available online at http://www.backupcentral.com. The one challenge, of course, is that these chapters have not been updated. Therefore, we've put them in our wiki so that anyone who wants to help us update them can do so.

- Tru64 Bare-Metal Recovery
- IRIX Bare-Metal Recovery
- Informix Backup and Recovery
- Clearcase Backup and Recovery
- High Availability

Speaking of BackupCentral.com

We've completely redesigned http://www.backupcentral.com using a content management system, forums, and MediaWiki. My number one goal for the new Backup Central is to make it much easier to provide you dynamic content and to build a strong community around backup and recovery issues. The new Backup Central has some really great features:

- phpBB forums for various backup-related topics, including one for discussing the book. Come join the discussions.
- A mailing list for each forum, allowing you to follow the discussions via the forum or email. Any posts in the forum are sent to the mailing list, and emails sent to the mailing list result in posts or replies in the forum.
- A multidirectional connection between backup-related Usenet newsgroups, mailing lists, and phpBB forums. One of the things I was reminded of while writing this book is that Usenet is alive and well, and I want to bring this great resource to the Backup Central community and to create another portal into this underutilized resource. Each relevant Usenet newsgroup has an associated mailing list and forum, and all messages to Usenet, the mailing list, or the forum go to the appropriate forum, mailing list, and newsgroup.
- A wiki based on MediaWiki, the same software that drives Wikipedia. One of the things you will find there is a wike entry for every chapter in this book. We're going to use these entries to update and further the

ideas you find in this book. We're doing this for two reasons:

- The first reason is that one of the problems with writing a technical book is that the second you go to press, something changes. While we're in the process of getting this book printed, MySQL will come out with three more storage engines, QTParted will probably support NTFS, and Bacula's Windows Server will become generally available. We'll use the wike to keep things like this up to date.
- The second reason is because my contributors and I don't know all the answers, folks. We did our best to come out with a solid book for you, but we haven't seen everything you've seen. We'd love it if you help us further the ideas mentioned in this book, help us to explain the scenarios under which a given procedure won't work, or how a given procedure should be enhanced. (For example, right now a friend of mine is trying to help me understand how to get `rsync` to better handle millions of files. His testing won't be done in time for press. Put it in the Wiki, Jason.) Join the new Backup Central and save the world—or at least its data.

I'll see you at http://www.backupcentral.com!

Conventions Used in This Book

The following typographical conventions are used in this book:

`Constant width`

: Indicates command-line computer output, computer-generated messages, and code examples. It is also used when referring to commands and parameters in text.

`Constant-width italic`

: Indicates variables in text.

`Constant width bold`

: Indicates user input in command-line examples.

`Constant width italic bold`

: Indicates variables in command-line examples.

Italic

: Introduces new terms and indicates URLs, files, directories, hostnames, and file extensions.

How to Contact Us

We have tested and verified all the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O’Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international/local)
707-829-0104 (fax)

We have a web page for the book that lists examples, errata, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596102463/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O’Reilly Network, see the O’Reilly web site at:

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O’Reilly Network Safari Bookshelf.

Safari offers a solution that’s better than e-books. It’s a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at http://safari.oreilly.com.

This Book Was a Team Effort

It's true; my license plate does say MRBAKUP. But that doesn't mean I know everything about backup and recovery. In fact, I've never even used some of the operating systems or database platforms covered in this book! It would be a disservice to you, the reader, for me to write chapters on those products—but I wanted the chapters in the book. So I hired a team of experts to write the chapters for you. Approximately 250 pages of this book were written by others, and contributors are recognized at the beginning of the chapter(s) they wrote.

Contributors

It's not an easy thing to write a chapter in someone else's book. Not only do you have to write, but you have to write based on someone else's design. There are also tight deadlines, and the process is nothing but hurry up and wait. I couldn't have done it without them, so please allow me to formally thank all of my contributors.

Amanda

: Contributed by Dmitri Joukovski and Stefan G. Weichinger. Thanks for seeing this one through.

BackupPC

: Contributed by Don "Duck" Harper. Quack!

Bacula

: Contributed by Adam Thornton. Thanks for bringing Bacula to the book.

Near-CDP

: Contributed by Michael Rubel, Ben Escoto, and David Cantrell. This chapter morphed a few times, and I

appreciate your patience as it gelled in my head.

AIX bare-metal recovery

Contributed by Mark Perino. I think you are the fastest writer on the team.

HP-UX bare-metal recovery

Contributed by Eric Stahl and Ron Goodwyn. Great collaborative effort, guys.

Linux and Windows bare-metal recovery

Contributed by Reed Robins. Maybe we can do it this way, or that way, or that way! Did I change the scope of the chapter enough? Thanks.

Mac OS X bare-metal recovery

Contributed by Leon Towns-von Stauber. Thanks, Leon. I sure am glad Mario told me to give you a call. Your chapter was perfect.

Solaris bare-metal recovery

Contributed by Aaron Gersztoff. Be careful what you ask for, right, Aaron?

DB2 backup and recovery

Contributed by Jeff Richardson, Kulvir S. Bhogal, and Kondal Yennaram. You guys all came through in a pinch, and I’m most grateful.

Exchange backup and recovery

Contributed by Scott Harris. More pictures! Fewer pictures! Make it like this, no make it like that! Isn’t it fun writing for me?

SQL Server backup and recovery

Contributed by Scott Harris. Look at that, Scooter! You’re the only one who was crazy enough to write two chapters for me. Thanks.

Sybase backup and recovery

> Contributed by Edward Barlow, who updated a chapter originally written by Bryn Smith. Another contributor I couldn't have done without. Thanks.

Dump internals

> Contributed by David Young. When are you going to move out here with your Mom?

Without these folks, this book would contain substantially less information than what you find here.

Technical Editors

Another group of people I must thank is our technical reviewers, and we had a *lot* of them! The problem with writing a book with a scope this big is that you need specialists and tech reviewers as well. Because of that, most technical editors reviewed only one or two chapters. I couldn't have done it without them. I'm sure I've miss a few, but here's my best attempt at listing them all (alphabetically by first name):

> Adrin Kow
> Andy Shellam
> Anthony Johnson
> Axel Schwenke
> Ben Garrett
> Brian Eliassen
> Brian Peasland
> Charles Whealton
> Chris Thomas
> Christoph Haas
> Craig Barratt
> D.A. Morgan
> Dana Diederich
> Daniel Callahan
> Dave Mehler

David Boyd
Edward Conba
Eric Gilmore
Eric Stahl
Finn Henningsen
Frank Sweetser
Greg Lehey
Ian Gorrie
Ian Herd
James Bougor
Jayesh Thakrar
Jeff Badger
Jeff Frost
Jeff Harbert
Jeff Richardson
Jeffrey P. Humes
John Haight
John Hurley
John Madden
Kern Sibbald
Kumar Sundaram
Lenz Grimmer
Marcel Lans
Mark D. Powell
Mark Dawson
Mark Perino
Massimiliano Daneri
Matthew Huff
Megan Restuccia
Mike Harrold
Mohammed Mehdi
Neal A. Lucier
Norbert Munkel
Patrick Matthews
Paul Muggeridge
Ralph R. Hirtler
Rob Worman

Rodrigo Real
Satyaprakash Pandey
Scott Boss
Scott Harris
Shane Seymour
Simon Riggs
Steve Hanson
Stewart Smith
Tammy Bednar
Todd Toles
Víctor A. Rodríguez
Vitalis Jerome
Wil Coulbourn
William Cole

Horror Stories

Also giving this book some flavor are those who contributed horror stories. Even if I couldn't use your story in the book, I want to thank you for sending one in.

Brian O'Neill
Brian Sakovitch
Chris Pritchard
David Bregman
David J. Young
Harry Tirrell
Hywel Matthews
Jack Coats
James Hunt
Jason Frankovitz
Jason Shupe
Jim Damoulakis
Jim Donnellan
John Merryman
Jorgen Lie
Karl Langdon

Kevin Suttle
Mark Perino
Michael Rice
Michael Tobin
Natalie Meek
Richard Ackerman
Scott Boss
Theo Van Dinter
William Birch
William S. Duncanson

Special Mention

There were a few people who were extremely helpful in one way or another throughout this project. I'd like to send a special thank you to them.

Anthony Johnson

> It's not every CEO who would volunteer to tech review a chapter that is essentially a free competitor with his own product. I hope you and Storix do very well. You've got quite the Linux and AIX bare-metal recovery tool there.

Brian Peasland

> You beat the living crap out of the first draft of the Oracle chapter, and rightfully so. The new chapter is *significantly* better thanks to your thorough and honest review. (You made me rewrite half of it, dude!)

Deb Cameron

> Isn't it fun editing a book with 18 authors from 3 continents, several time zones, a few different native languages, using about 60 technical editors? Let's do it again sometime!

Joshua D. Drake

Thanks, Joshua for spending the time you did with me on the phone to help me better understand PostgreSQL. Next time, you should be more forthright with your own opinions. A guy really doesn't know where he stands with you.

Lenz Grimmer

You were my liaison in the MySQL community, and I definitely needed the help. Thanks to you and the whole MySQL team.

Lynn Stone

Thank you for helping get this project off the ground. I couldn't have done it without you. Only I know your secret identity.

Tammy Bednar

For such a busy woman, you gave me exactly what I needed for the Oracle part of this project. You've obviously done a lot of work on the Oracle backup products, and it shows. I hope you'll see my newfound respect for your products in the Oracle chapter.

Zmanda

Thanks for the support on the Amanda chapter, and for bringing commercial support to a very popular open-source backup tool.

Mario Obejas

Thank you so much for referring me to Leon.

I Don't Know It All

If there's one thing I learned while writing this book, it's that I do not know everything there is to know about backups. If you have a better way to do anything described in this book, have learned any special tricks, or have written any neat

utilities that you think would help other people do backups and recoveries, let me know. Email me at curtis@backupcentral.com. Your tricks or utilities may be included in the next edition of the book and listed immediately on http://www.backupcentral.com.

How Can I Say Thanks?

How can I begin to thank the hundreds of people who helped me?

To God: May any praise for this book go to You alone.

To my wife, Celynn: I say “thank you” for the many nights you spent alone while I pounded away at my keyboard somewhere around the globe. You’re a special woman who never gave up on me or my dream. I love you.

To my daughter, Nina: You were only seven when the first book came out. Now you’re a beautiful young lady who is growing up so fast. I’m going to have to get a gun and sit on the porch.

To my daughter, Marissa: You were only two when the first book came out. Now you’re a beautiful nine-year old—my, how time has flown. Let’s go to the park and ride our bikes together.

To my parents: What can I say? You always believed in me. You always used to tell me, “I don’t care if you’re a ditch digger. Just be the best darn ditch digger in the world.” Well, being a backup guy is as close as you can get to being a ditch digger in the computer business, and I “wrote the book” on that.

To Bob Walker for helping me get my first job in backups, and Ron Rodriguez for being all too eager to give it to me.

To Susan Davidson, who didn’t fire me when I couldn’t recover that purchasing database in 1992: that second chance was all I needed to become the expert in backup that I

am today. If you had fired me (and I'm sure a few people wanted you to), who knows where I'd be today. (If you're curious about the story, look for the sidebar "The One That Got Away" in Chapter 1.)

To Collective Technologies for helping me round out my skills enough to see that I wanted to specialize in backup and recovery, and for supporting me when I wrote the first book.

To Jason Stege, Robin Young, Jeff Williams, Reed Robins, and Elia Harris: Thanks for believing in me when I started my own company. I hope I did right by you.

To Mark Shirman and all my friends at GlassHouse: Thanks for giving me a place where I finally feel like I'm using my talents.

To my wife's family: Thank you for raising such a wonderful lady. Thank you for treating me as one of your own and supporting us on our quest. *Pahingi ng sinagang?*

To all the teachers who kept trying to get me to live up to my potential: You finally got through.

To O'Reilly: Thank you for the opportunity to bring this much-needed book to market.

To Deb Cameron and Michael Loukides, my editors: We'll have to actually meet one of these days! I don't know how you do this, reading the same book over and over, without letting your eyes just glaze over. You're great editors, and I could really tell that you put your all into this project. Thank you, thank you, and thank you. (Now don't edit *that* sentence, OK?)

To the reader: Thank you for purchasing this book. I hope you learn as much reading it as I did writing it.

Part 1. Introduction

Part I consists of the following two chapters:

Chapter 1, The Philosophy of Backup

Describes why you should back up, and a little bit about how to do it.

Chapter 2, Backing It All Up

Goes into detail about the essential elements of a good backup and recovery system.

Chapter 1. The Philosophy of Backup

I back up; therefore, I will be.

When I look at the title of this chapter, I think about the old Steve Martin stand-up routine in which he said that in philosophy class, “you learned just enough to screw you up for the rest of your life.” (Steve studied the important questions, like “Is it OK to yell ‘movie’ in a crowded fire house?” I promise not to do that.) However, “The Philosophy of Backup” did seem like an appropriate name for this chapter, since we’re going to talk about the *why* of backup. (We’ll also talk a little about the *how*, of course.)

Champagne Backup on a Beer Budget

A good backup and recovery system is essential for a company of any size. Unfortunately, IT doesn’t always get the budget it needs, and the backup system almost never gets the money that it needs. Well, if you agree that you need a very good backup system, but you don’t have enough money to pull that off, know that this book was written with you in mind. You need champagne backup on a beer budget. Welcome to the club.

Just because you have a small budget doesn’t mean you have to do without backup. Most of the backup systems in this book can be implemented in small environments for a few hundred dollars—including hardware.

Tip ✎

Don’t worry, enterprise customers—there’s plenty in here for you as well. The more you use the techniques taught in this book, the more money you can save for other IT projects. By the time you’re done implementing all the ideas in this book, hopefully my next book will be done, which will be right up your alley. It will cover nothing but commercial data protection solutions, including multiplatform commercial backup and recovery systems, continuous data protection, near continuous data protection, data de-duplication backup systems, replication, and the like.

Now that you’ve read this far, you may find yourself asking questions like these:

- Why should I read this book?
- Can I really back up with open-source backup software?
- Why should I be using disk?
- Why should I back up at all?
- How do I find a balanced way to back up (wax on/wax off)?

Let’s get started answering these questions.

Why Should I Read This Book?

If you've been doing system administration for some time, you may be asking yourself this question. There are many answers. Perhaps self-preservation is your primary motivator. You'd like to make sure you don't lose your job the next time a disk drive dies. Perhaps you've already got a decent backup system and you'd just like to make it better. Maybe you are looking for some new ideas about how to deal with upcoming backup and recovery needs. What follows are some of the reasons *I* think you should read this book.

Schadenfreude

Schadenfreude is a German word that means to take joy in the misfortunes of others. It's why we watch those weird videos on the Internet where some idiot tries to do something stupid and ends up hurting himself. Each of the sidebars in this book is a true horror story that really happened to someone I know. These are not urban legends or horror stories passed on from admin to admin. These are firsthand encounters with disaster. There's a schadenfreude element to reading these stories, of course. But each story also makes a point, and it was not just made up to make that point. The things that I warn about in this book really happen. This can be a very tough job if you are not prepared, so read closely. You might want to start by reading the sidebar "The One That Got Away" later in this chapter. It's the story of the defining moment in my career.

The One That Got Away

"You mean to tell me that we have absolutely no backups of *paris* whatsoever?" I will never forget those words. I

had been in charge of backups for only two months, and I just knew my career was over. We had moved an Oracle application from one server to another about six weeks earlier, and there was one crucial part of the move that I missed. I knew very little about database backups in those days, and I didn't realize that I needed to shut down an Oracle database before backing it up. This was accomplished on the old server by a `cron` job that I never knew existed. I discovered all of this *after* a disk on the new server went south.

"Just give us the last full backup," they said. I started looking through my logs. That's when I started seeing the errors. "No problem," I thought, "I'll just use an older backup." The older logs didn't look any better. Frantic, I looked at log after log until I came to one that looked as if it were OK. It was just over six weeks old. When I went to grab that volume, I realized that we had a six-week rotation cycle, and we had overwritten that volume two days before.

That was it! At that moment, I knew that I'd be looking for another job. This was our purchasing database, and this data loss would amount to approximately two months of lost purchase orders for a multibillion-dollar company.

So I told my boss the news. That's when I heard, "You mean to tell me that we have absolutely no backups of *paris* whatsoever?" Isn't it amazing that I haven't forgotten its name? I don't remember any other system names from that place, but I remember this one. I felt so small that I could have fit inside a 4 mm tape box. Fortunately, a system administrator worked what, at the time, I could only describe as magic. The dead disk was resurrected, and the data was recovered straight from the disk itself. We lost only a few days' worth of data. Our department had to send a memo to the entire company

saying that any purchase orders entered in the last two days had to be reentered. I should have framed a copy of that memo to remind me what can happen if you don't take this job seriously enough. I didn't need to though; its image is permanently etched in my brain.

Some of this book's reviewers said things like, "That's pretty bold! You're writing a book on backups, and you start it out with a story about how you messed up. Some authority you are!" Why did I include it? Through all the years, and all the outages, this one sticks in my mind. Perhaps that's because it's the only one that almost "got me." Had it not been for the miraculous efforts of a wonderful administrator named Joe Fitzpatrick, my career might have been over before it started. I include this anecdote because:

It's the one that changed the direction of my career.

There are several valuable lessons that I learned from it, which I discuss in this book.

It could have been avoided if I had had a book like this one.

You must admit that it's pretty darn scary.

You Never Want to Say These Words

"We lost only a few days' worth of data." In the sidebar "The One That Got Away," I said that we lost only a few days worth of data. I swore the day I said these words that I would never say them again. From that day forward, I was convinced of the importance of backups. I never again assumed anything, and I began to study everything I could about backup technology. This book represents my attempt to compile what I have learned about inexpensive backups into a single volume, and it is written so that no one who

reads it should ever need to utter the preceding statement. In my opinion, *no amount of data loss is acceptable* . I would also wager that you would be hard-pressed to find an end user who would feel much different. Whether it's a spreadsheet that one person created or a customer database representing hours or days of sales invoices and the efforts of hundreds of people—ask the person who needs the data how much data loss they think is acceptable. Every statement, every opinion, every story, and every chapter in this book is based on the premise that any data loss is unacceptable. Let me state that again for emphasis.

> **Tip ✎**
>
> With the technology that is now available, there is no reason for any data to be lost—that is, *if* backups are given the proper attention and priority that they need.

You're Curious About Open-Source Backup Products

Just a few years ago, you could perform your backups with a few scripts and `dump`, `tar`, or `cpio`, or `ntbackup`. The demand for midrange computers grew astronomically, and the need for bigger databases, larger drives or filesystems, long filenames, and long pathnames grew proportionally. These large databases and filesystems started shipping, which then created a large market for commercial backup utilities, and one or two such products emerged; scores of others eventually followed.

Some of these early products were just GUIs and volume management built on top of existing native backup utilities to provide enhanced levels of functionality. Other

companies felt that these native utilities had many limitations that could not be fixed without abandoning them altogether. Those companies chose to develop custom, even proprietary, backup methods. They attempted to overcome the limitations that products based on `dump` and `tar` could not overcome.

In recent years, the demand for centralized backup and recovery has also given rise to a number of open-source backup and recovery tools, six of which are covered in this book. The open-source backup market followed a pattern similar to the commercial products mentioned. The original open-source backup product, Amanda, is a wrapper around the native utility of your choice. BackupPC leaves data in its original format, and Bacula uses a custom format designed to overcome the limitations of GNU `tar`.

There are now a number of choices in the open-source backup market. It's quite possible that one or more of the open-source products covered in this book can meet your backup and recovery needs. This book is currently the only resource that covers all of these tools in a single place.

You Want to Learn About Disk-Based Backup

If you haven't heard of disk-based backup or disk-to-disk-to-tape (D2D2T) backup, then it's time to turn off the digital video recorder (DVR) and pick up a trade magazine or two. (Of course, your DVR is nothing more than disk-based backup of your TV. And if you're occasionally making VHS tapes of your DVR shows, it's even a D2D2T system.) The use of disk in backup and recovery systems has exploded in the last few years, and it's really solving a lot of problems.

Chapter 9 covers backup hardware and goes into much more detail about why disks have become a very attractive

backup target. Here is a quick summary of some of those reasons:

Cost

The biggest reason that disk has become such an attractive backup target is that the cost of disk has been dramatically reduced in the last few years. The cost of a reasonably priced disk array is now approximately the same price as a similarly sized tape library filled with media. When you consider some of the things you can do with disk, such as eliminating full backups and redundant files, disk becomes even less expensive.

Reliability

Unlike tapes, disks are closed systems that aren't susceptible to outside contaminants. In addition, the actual media of a hard drive is, well, *hard* when compared to a piece of tape media. The result is that an individual disk drive is inherently more reliable than a tape drive. Disk drives become even more reliable when you put them in a RAID array.

Flexibility

Generally speaking, tape drives can only go two speeds: *stop* and *very fast*. Yes, some tape drives support variable speeds. However, they can usually only slow down to about 40 percent of the rated speed of the drive. Disk drives, on the other hand, work at whatever speed you need them to go. If you need to go a few hundred megabytes per second, put a few drives in a RAID group, and blast away. Then if you need that same RAID group to write at 10 KB/s, go ahead. Unlike tape drives, disk drives have no problem writing slowly, then quickly, then slowly, then.... You get the picture. This makes disk a perfect match for unpredictable backup streams. Once all that random data has been written in a serial fashion

on your disk device, the disks can easily stream backup data to tape—if that's what you want to do. Some people are foregoing that step altogether and replacing it with replication. Try doing that with a tape drive.

Disk-based backups are also an extremely economical way to bring completely automated backups to small and medium businesses (SMBs). While a large tape library can be very inexpensive (on a dollars-per-gigabyte basis) and very expandable, the same is not always true of smaller libraries aimed at the SMB market. The big challenge is expandability. The less expensive a tape library is, the less expandable it usually is. (There are always exceptions, of course.) By comparison, some of the completely automated open-source backup products mentioned in this book can be used with a single disk drive costing less than $100. If you need to expand beyond that, just buy another disk and add it to your volume manager. You can also buy RAID controllers that allow you to start with one disk and add more as your needs grow. You can use this method to expand from hundreds of gigabytes to many terabytes of capacity.

Why Back Up?

I've heard it all. I've been accused of caring only about backups. It's been said that I think the whole world revolves around a cartridge reel. I've said that someday the world's going to crash, and I'm going to have the backup. The question is: how serious are *you* about protecting your data? To help you come to a decision on this matter, let's talk about what happens if you don't have good backups.

That's Not What I Meant!

I was administering a QA group for a major software company. When the software did its setup, it created a directory in *$HOME/foo* so the software could install the user portion. A QA person was doing the install under root. The software created the directory *$HOME/foo*; literally, *$HOME* was the directory name. He submitted the bug fix, and decided to get rid of the useless directory. You've probably guessed it:

```
# rm -rf $HOME
```

(This was on a standard Unix system where *$HOME* for root was still /.)

Once I finally stopped laughing, I went and got the install media to rebuild the machine (no jumpstart image for that one, unfortunately, nor any backups for the various QA servers). Fortunately, most of the critical data was on the NFS server.

William Birch

What Will Lost Data Cost You?

To answer this question, you need to consider what kind of data you are backing up. This is a perfect time to include people who may not consider themselves computer people. Get input from other departments to answer this question. When all those 1s and 0s come together, just what kind of information are we talking about? Do you use manual accounting methods or are your company's financial records stored in some accounting software somewhere? When a customer calls in and orders something, do you jot that down on a carbon-copied order form or do you enter it in some sort of order processing program? What about things like budgets, memoranda, inventories, and any other "paperwork" that you throw around from day to day? Do you keep copies of every important memo that you send, or do you depend on the computer for that?

If you're like most people, you have grown quite dependent on these things we call computers. You forget how much of your work has been saved in the form of little magnetized bits spread out across a bunch of spinning platters. Maybe you work in an environment in which you've never lost a disk, so you've never had to do a restore. Maybe you've never fat-fingered a key and deleted an important file. If that's the case, remember what my dad used to say: "motorcycle riders come in two types—those who have fallen and those who will fall." The same is true of disk drives. If you've never had a failed disk drive, trust me, your turn is coming!

So what would you lose if you lost data? To quantify this, we need to examine the types of information that may reside in your environment and what would happen if you lost each type of information. Most of what you could lose is very tangible—and quantifiable in monetary terms—and it might surprise you.

Lost customers

This is quite possibly the most tangible and most devastating of all losses. If your entire customer database is on a computer somewhere, how will you know who they are if that computer dies? You might actually “lose” your customers and never find them again. You could also lose customers who depend on data that is on one or more of your computers; if the customer finds out that you have lost his data, he will undoubtedly be less than impressed with you. The degree to which this data loss affects him may not even be relevant to him; he knows that you lost his data, and he might leave just because he no longer feels your company is competent.

Orders

Whatever service or product your company provides, you have some way to keep track of requests for that product or service. Again, chances are that the method is computer-based. Data loss may mean several hours, days, or even weeks of lost orders. These may be orders that your salespeople worked very hard to get!

Morale

Think about how you would feel if you were one of the salespeople whose orders were lost. You spent days or weeks working on sales, and now they’re gone forever. Maybe you should go somewhere where your hard work doesn’t go to waste. The better the salesperson, the better the chance that she may jump ship if you lose her sales. What about the average employee? If your computers have a reputation for going down and a reputation for losing data, it gives the employees a feeling of helplessness. Maybe they should go somewhere where they have the proper equipment to do their jobs.

Image

What about your standing in the industry? News of a major data loss undoubtedly spreads. This news may get to competitors, whom you can trust to use it against you at any opportunity. The news may also get to a regulatory agency that is in charge of your type of company. For example, if you work for a U.S. bank, it would be a terrible thing for the Office of the Comptroller of the Currency (OCC) to find out that you had a major data loss. They may decide to take a really close look at your affairs. Nobody wants that kind of attention!

Budget

It takes only one story of lost data to give your computer department an internal reputation for data loss. Try as you might to get rid of it, that reputation may stay for a while. You're only as good as your last restore. (A friend of mine said, "You're only as good as your *worst* restore.") If people don't trust your backups, they will duplicate your backup efforts. Employees will spend time and money backing up their systems locally. Each person may decide to buy his own backup drive and backup software or even to come up with his own in-house script. Their backups will be inefficient and costly at best, and may subject them to further data loss at worst. When everybody takes matters into their own hands, you can lose quite a bit of money in people-hours and extra hardware.

Time

How many people are supporting your computers? How much of their efforts will you lose if your development system loses data? I know of many companies that have numerous contract programmers writing code all the

time. If the system that houses their work loses their code, how much money will you have wasted? In fact, no matter what department you look at, if they do their work on a computer, and you lose that data, you can lose considerable time and money.

What Will Downtime Cost You?

When planning your backup and recovery program, you may have several options that affect the speed of the recovery. The faster the recovery, the more the backup system will cost you. What you must ask yourself before deciding on these types of options is, “What will downtime cost?” When thinking about this, I’m reminded of a copier machine commercial from a few years ago: “When your copier goes down, do people just say, ‘That’s all right, we’ll just use carbon paper!’?” If one of your main systems goes down, can your people continue working, or does your entire company come to a standstill? If it comes to a standstill, are your people salaried, so that sending them home saves you no money? Here are some additional costs to consider:

Customer perception

A customer hates to hear, “Please call back; our computers are down,” or “Connection not responding.” Depending on your type of business, they might just decide to go elsewhere. The longer your systems are down, the more customers will hear this message.

Employee perception

Nobody wants to work at a company where the computers are always going down. The more your employees depend on your systems, the truer this becomes. If you were a salesperson who couldn’t use

your contact database for a day or so, how happy would you be?

Time

Again, you lose time. You lose headway, and your salaried employees who depend on the system that is down are effectively being paid to do nothing.

Wax On, Wax Off: Finding a Balance

Using a system that has no backups is like driving a car 100 miles an hour down a busy road the day after your insurance policy expires. Likewise, having a three-node, highly available cluster for a noncritical application is like having full coverage on your 20-year-old fifth car. Just as insurance plans have different levels of coverage and riders to cover various types of damage, different backup methodologies provide different levels of recoverability.

That Was Close

One memorable moment was when we had a 600 GB file server that hadn't been properly backed up in a while. During a particularly hot weekend, both A/Cs handling the room failed, and temperatures soared. We shut everything down, waited for the A/Cs to be fixed, and started things back up after it cooled off a bit. Sure enough, two disks, physically next to each other in the same RAID4 array, failed. We were narrowly able to avoid total data loss by finding a spare disk and swapping control boards between it and one of the failed drives, which let it spin up and be accessed. We had the vendor courier us replacement drives the next day and then spent a lot of time fixing the backup server.

Theo Van Dinter

Don't Go Overboard

Not all environments need up-to-the-minute data recoverability. For many environments, recovering the systems up to last night's backups is acceptable. For some environments, recovering the system even up to last week

or month is OK. Spending thousands of dollars and hundreds of hours implementing the greatest backup solution in the world is a waste if you don't need that level of coverage. This usually is not the problem for most sites; on the contrary, most sites don't spend nearly enough money or effort on their backup and recovery systems. In other cases, however, money may be wasted on unnecessarily elaborate systems.

Recoverability requirements also vary from machine to machine within the same company. The amount of work that would be lost, or the possibility of adversely affecting a customer, may determine these requirements. For example, it may be considered acceptable for an employee or two to lose a day's work spent on a few word processing documents. That is, unless it was the Senior Vice President's assistant who was working on the departmental budget, in which case your mileage may vary. And, it would probably be totally unacceptable for you to lose even one hour's worth of entries into a companywide sales database used by hundreds of people.

The point is that *your backup requirements are determined by your recovery requirements*. The difficulty comes in finding and using a tool capable of providing the level of recovery that you need. Consider users' home directories for a minute. If they are local to each user's workstation, a loss of one user's disk in the afternoon would mean that one user would lose a few hours of work. However, if user directories are located on an NFS file server that serves thousands of users, you could potentially lose several thousand hours of work if you use only traditional backup tools.

Tip ✎

If the loss of a networked file server is unacceptable, you might want to consider *snapshot* technology. Snapshot software allows you to take a “picture” of your drive or filesystem at a single point in time and then use that picture to back up that drive or filesystem. If the backup references the drive or filesystem via this snapshot, it will back up a consistent picture of the drive or filesystem as it looked at the time the snapshot was taken. If this kind of functionality is interesting to you, you might consider reading Chapter 7, which describes emulating snapshot functionality with `rsync` and hard links.

Sometimes the tool you need comes with your operating system or database platform, but it’s just not being used properly. Sometimes backup tools aren’t being used at all. For example, if you have a production Oracle database, combining nightly hot backups with archived redo logs provides you up-to-the-minute recoverability. However, if you lose a disk that is part of a database that doesn’t back up its transaction logs, you will lose all work since the last cold backup. See Part V for more information .

Warning ☠

If you have a production instance of any kind and are not using the transaction logging feature of your database engine, turn on logging as soon as possible!

Therefore, while it is necessary to find the appropriate utility to give you the degree of recoverability that you require, it is also necessary to use it.

Get the Coverage That You Need

Some environments cannot afford even one minute of downtime, and they should pay for the best backup coverage—whatever it costs. This is because of the great loss that they will incur if they ever lose their systems for even a short period (I know of one company that claims that it loses over $1 million a minute when its systems are down). On the other hand, if you are in an environment that can afford downtime, then spending huge amounts of money for an immediately available *hot site* [1] is a complete waste of money.

Consider Table 1-1. No one should depend on a car, or a computer, without having at least the basic level of coverage. If the only car that you own is uninsured and a drunk driver runs into you and totals it, how would you recover from such a loss? Similarly, if your computer systems have critical information stored on them, how will you recover when a hard drive crashes and all that data is lost? What some people forget is that the opposite of this equation is true as well. If you have a third car that happens to be a 20-year-old (nonclassic) car, you will probably get only liability coverage on it; you could live without that car if it were destroyed today. Spending hundreds of extra dollars a year to insure a $50 car just doesn't make sense. Likewise, if the computers that you are managing are in an environment in which you can do without them for a few days, do you really need hot-swappable, mirrored drives? Pick an appropriate level of protection for your environment.

Table 1-1. Comparing automobile insurance and data protection

Types of coverage	Automobile insurance	Computer backups

Types of coverage	Automobile insurance	Computer backups
Minimum coverage	Collision and liability (just keeps you from losing your shirt if you run into someone).	Regular nightly backups (keeps you from losing your job when a disk drive dies)
Unexpected disasters	Comprehensive coverage (vandalism, acts of God, etc.).	Journaling filesystemsUninterruptible Power Supplies (UPSs)
Get me driving now	Rental car coverage (you get a car if your car is in the shop due to an accident).	RAIDMirroringUsing hot-swap drivesHigh-availability (HA) system
Major disasters	Another company will pick up your policy and replace your car if both your car and your insurance company are destroyed in an earthquake.	Sending copies of your backup volumes to off-site storage, in case both your computer room and media library are destroyedSending your backups via a dedicated network to a large storage system at your off-site storage vendor
Maximum protection	The insurance company not only agrees to the conditions listed earlier, but also agrees to store another car of the same model in another state that you can use at any time if all cars in your state are destroyed.	Real-time mirroring to a hot-swappable system at another of your sitesSending your backups via either network or courier to a hot-site vendor

You need to balance the cost of a particular backup implementation against the projected monetary loss of the outage from which it protects you. For example, assume that you are evaluating two backup choices. The first option involves sending copies of your backup volumes to an off-site vendor for storage at a cost of $500 a month. The second option is an immediately available standby machine in another city that receives up-to-the-minute replication data from your production machine; let’s say this option costs you $5,000 a month.

Your company is located in Utopia, where no natural disasters ever occur, your disks are all mirrored, and you have determined that a day’s worth of downtime would cost

only $500. Do you really want to spend $60,000 a year to protect against something that will probably never occur? If something catastrophic happened to your datacenter, wouldn't the day-old, off-site copies serve just as well? Your company would suffer an extra day or so of downtime, but you have already determined that this is affordable. The $6,000-a-year solution is probably much more appropriate for this environment.

However, are you protecting yourself from everything that you should be? Are you in an area that is prone to natural disasters and yet have no protection against that sort of event? Maybe you need to consider a different type of off-site storage. If you have a customer base that needs the data on your computers on a regular basis, have you provided for quick recovery in case of a failure? Perhaps you should be considering a hot site or multiple-site mirroring of your database servers. Table 1-1 provides a good overview of the various levels of coverage.

Why the Word "Volume" Instead of "Tape"?

Most backup utilities were originally written to back up to tape. Therefore, most books and online manuals talk about backing up to tape. However, many people are backing up to CDs, magneto-optical disks, or even disk drives. These media types have many advantages, because they act more like disk drives than tape drives. Random access of backup data is easier and you can read them using any block size you wish, because they do not record interrecord gaps like tape drives do.

Since many people no longer use tape, this book uses the more generic word volume whenever appropriate. You'll also find the term backup drive instead of tape drive. Again, that is because the backup drive could be a CD

burner or a disk drive. The book uses the words *tape* and tape drive only when they are necessary and appropriate.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

---

[1] A hot site is a place where you have computers standing by to do an immediate recovery of your environment.

Chapter 2. Backing It All Up

Now that the philosophy lessons of Chapter 1 are over, it's time to look at some of the important concepts behind backup and recovery, such as what to include, when to back it up, and more.

Don't Skip This Chapter!

The casual reader might assume that this chapter is an introduction to basic backup concepts. While that is, in fact, the purpose of this chapter, it is also true that many seasoned administrators are unfamiliar with the ideas presented here. One reason for this is that administrators find themselves constantly being pulled away from "mundane" activities like backups for things that are thought to be more "important," such as installing new servers and figuring out why systems are running slowly. Also, administrators may go several years without ever needing to perform a restore. The need to use your backups on a regular basis would undoubtedly change your ideas about their importance.

I wrote this book because backup and recovery has been my primary area of emphasis for several years, and I would like to share the lessons I've learned from this focused activity. This chapter provides an overview of how your backups should work. It also explains many basic yet extremely important concepts upon which any good backup plan should be based, and upon which all implementations discussed in this book are based.

The Impossible Job That No One Wants

Would anyone reading this book say that losing data is OK? I don't believe so. Then why do we treat backups so lightly? Sometimes I feel like Rodney Dangerfield when I'm arguing for better backups—"I tell ya, I don't get no respect, no respect." Backups often aren't considered during systems design. When a new server is purchased, does anyone ask for the impact on the current backup methodology? Some IT departments do not even have control over the purchase of new systems, because they are sometimes bought by other cost centers. Have you ever tried to explain to another department manager why his terabyte-sized database server isn't going to get backed up to the standalone, gigabyte-sized tape drive that came with it?

Another often-overlooked issue is backup personnel. Have you ever tried to find the person in charge of backups? It's often an extra duty that gets passed around, in a manner similar to the way my sister and brother and I argued over whose turn it was to wash the dishes. If you are lucky enough to have a dedicated person, it's usually the most junior person in the company. I know, because that's how I got my first job. In fact, that's how many people get their first jobs. How can we give such low priority to something so important? Perhaps we should change that. Will one book change this long-standing hiring tradition? Probably not, but maybe it will help. At the very least, if the person in charge of backups has this book, that person has a complete guide to accomplishing the immense task that lies ahead.

What's the big deal, you say? With modern computer systems and reliable disk drives, why are backups still so important? Because computers still go down, that's why. Also, companies are placing more reliance than ever on computers functioning reliably. I don't care how good your Unix vendor is or how reliable your disk drives are or even if you have Dogbert himself as your network administrator,

systems go down. Murphy's Law thrives in computer systems. Not only will your computer systems go down occasionally, but they will do so at the time most inconvenient to you and your customers. At that moment, and that moment will come, it is the job of the backup person to replace the data on the disk or disks that have stopped the show. "How long will it take?" is a typical question. The only acceptable response is "it's already done."

Who wants to be the person who messed up the restore and caused the customer database to be offline for three extra hours? Who wants to be the person who has to send a memo to the entire company saying that any purchase orders entered in the last two days have to be reentered? Who wants to be the person who has that in mind every day as they are checking the results of last night's backups? If you do your job well, and no data is lost, you are just doing what you're supposed to do. If you mess up, you're in big trouble. Who wants that job? No one, that's who.

You're reading this book because you've got the impossible job that nobody wants. Whether you've been doing it for a while or have just started down the backup road, you can see that the task that lies ahead is immense. The volume of data is tremendous, the nature of the data changes constantly, and the utilities at your disposal never seem to be up to the job. I know because I've been there. I've spent months trying to implement "solutions" from operating systems and database products that weren't ready. I've seen companies spend money on expensive commercial utilities, only to buy the wrong utility for their application. I've watched newer and bigger servers roll in the door without a single backup drive among them. I've also spent long nights and weekends in computer rooms trying to recover data in a "reasonable" amount of time.

Unfortunately, “reasonable” is defined by the end user who has no idea how difficult this job is.

There are now solutions to almost every backup problem out there. If you run a small shop with just a few systems, all of which run the same operating system, there’s a solution for you. If you work in a huge shop with hundreds of boxes in the various flavors of Unix, Linux, Windows, and Mac OS, or just a few multiterabyte databases, there’s a solution for you. The biggest part of the problem is misinformation. Most people simply do not know what is available, so they either suffer without a solution or settle for an inferior one—usually the one with the best salesperson. The six important questions that you have to continually ask yourself and others are why, what, when, where, who, and how:

Why?

> Why are you protecting yourself against disaster? Does it really matter if you lose data? What will the losses be? What different types of data do you have, and what is the value of each type?

What?

> What are you going to back up, the entire box or just selected drives or filesystems? What operating systems are you going to back up? What else, besides normal drives or filesystems, should be included in a backup?

When?

> When is the best time to back up your system? How often should you do a full backup? When should you do an incremental backup?

Where?

Where will the backup occur? Where is the best place to store the backup volumes?

Who?

Who is going to provide the hardware, software, and installation services to put this system together?

How?

How are you going to accomplish it? There are a number of different ways to protect yourself against loss. Investigate the different methods, such as off-site storage, replication, mirroring, RAID, and the various levels of protection each provides. (Each of these topics is covered in detail in later sections of this book.)

Deciding Why You Are Backing Up

If you can’t answer this question, there’s really no point in moving forward. The good thing is that it is a really easy question to answer. Just think about all of the various things that can happen to your data, and then look at all of the types of data that you have. You should be familiar with each business unit that creates data, and how that business unit would be affected if that data was lost or damaged. All of this becomes your business justification for moving forward.

Deciding What to Back Up

Experience shows that one of the most common causes of data loss is that the lost data was never configured to be backed up. The decision of *what* to back up is an important one.

Plan for the Worst

When trying to decide what files to include in your backups, take the most pessimistic technical person in your company out to lunch. In fact, get a few of them together. Ask them to come up with scenarios that you should protect against. Use these scenarios in deciding what should be included, and they will help you plan the "how" section as well. Ask your guests: "What are the absolute worst scenarios that could cause data loss?" Here are some possible answers:

- An entire system catches fire and melts to the ground, leaving an unrecognizable mass of molten metal and blackened, smoking plastic.
- Since this machine was so important, you had it replicated to another node right next to it. Of course, that machine catches fire right along with this one.
- You have a centralized server that controls all backups and keeps a record of backup volume locations and what files are on what volumes, and so on. The server that blew up sits right next to this "backup server," and the intense heat takes this system with it.
- The disastrous chain reaction continues, taking out your DHCP and Active Directory servers, the NIS master server, the NFS and CIFS home directory

servers, and the database server where you house the inventory of all your backup volumes with their respective locations. This computer also holds the telephone database listing all service agreements, vendor telephone numbers, and escalation procedures.

- You haven't memorized the number to your new off-site storage vendor yet, so it's taped to the wall next to your backup server. You realize, of course, that the flames just burned that paper beyond recognition.
- The flames set off the sprinkler system, and water pours all over your backup volumes. Man, are you having a bad day....

What do you do if one of these scenarios actually happens? Do you even know where to start? Do you know:

- What volume contains last night's backup?
- Where you stored it?
- How to get in touch with the off-site storage vendor to retrieve the copies of your backup volumes? And once you find that out, whether your server and network equipment will be available to recover?
- Who to call to get replacement equipment at 2:00 a.m. on a Saturday?
- What the network looked like before all the wires melted?

First, you need to recover your backup server, because it has all the information you need. OK, so now you found the backup company's card in your wallet, and you've pulled back every volume they had. Since your media database is lost, how will you know which one has last night's backup on it? Time is wasting....

All right, you've combed through all the volumes, and you've found the one you need to restore the backup server (easier said than done!). Through your skill, cunning, and plenty of help from tech support, you restore the thing. It's up and running. Now, how many disks were on the systems that blew up? What models were they? How were they partitioned? Weren't some of them striped together into bigger volumes, and weren't some of them mirroring one another? Where's that information stored? Do you even know how big the drives or filesystems were? Man, this is getting complicated....

Validated, My Eye

A biotech firm with a number of servers that were considered validated systems for FDA CFR21 purposes lost a critical database that was running on one such server. When they went to their backup server to restore it, they discovered to their horror that that server had not been backed up for approximately three months. Somehow, it had been removed from the backup schedule, so no "errors" were showing up, and they were now without anything remotely approaching a current backup. The problem escalated up to the CEO of the company.

Jim Damoulakis

Didn't you just install that big jumbo kernel patch last week on three of these systems? (You know, the one that stopped all those network broadcast storms that kept bringing your network down in the middle of the day.) You did make a backup of the kernel after you did that, didn't you? Of course, the patch also updated files all over the OS drive. You made a full backup, didn't you? How will you restore the operating system drive, anyway? Are you really going to go through the process of reinstalling the operating

system just so you can run the `restore` command and overwrite it again?

Filesystems aren't picky about size, as long as you make them big enough to hold the data that you restore to them, so it's not too hard to get those filesystems up and running. But what about the database? It was using raw partitions. You know it's going to be much pickier. It's going to want */dev/rdsk/c7t3d0s7*, */dev/dsk/c8t3d0s7*, and */dev/dsk/c8t4d0s7* right where they were and partitioned just as they were before the disaster. They also need to be owned by the database user. Do you know which drives were owned by that user before the crash? Which disks were those again?

It could happen.

> **Tip ✎**
>
> Part IV covers these Catch-22 situations.

Take an Inventory

Make sure you can access essential information in the event of a disaster:

Backups for your backups

> Many companies have begun to centralize control of their backups, which I think is a good thing. However, once you centralize storage of all your backup information, you have a single point of failure for your entire backup plan. You can't restore the backup server because you don't have the database of your backups. You don't have the database of your backups because you need to restore your backup server. Restoring this server would be the first step in any multisystem outage.

For things like media inventory, don't underestimate the value of an inventory printed on paper and stored off-site. That paper may just get you out of this Catch-22. Given the single-point-of-failure factor, the recovery of your backup server should be the easiest and best-documented recovery that you have. You may even want to investigate creating a special `tar`, `ntbackup`, or `rsync` backup of that data to make it even easier to recover during a disaster.

What peripheral devices did you have?

Assuming you back up your disk drive configuration on a regular basis, you might have a list of all the disk drives, but do you know what models they are? If you have all Brand-X 500 GB drives, you have no problem, but many servers have a mixture of drives that were installed over time. You may have a collection of 40 GB, 100 GB, and 500 GB drives, all on the same system. Make sure that you are recording this in some way. Unix and Mac OS systems record this information in the *messages* file, and Windows stores it in the registry, so hopefully you're backing *those* up.

How were they partitioned?

This one can *really* get you, especially if you have to restore the operating system drive or a database drive. Both drives are typically partitioned with custom partitions that must be repartitioned exactly the same as before for a proper restore to occur. Typically, this partition information is not saved anywhere on the system, so you must do something special to record it. On a Solaris system, for example, you can run a `prtvtoc` on each drive and save that to a file. Search on the Internet for scripts for capturing this information; a number of such free utilities exist.

How were your volume managers configured?

> A number of operating system-specific volume managers are available, including Veritas Volume Manager, Windows Dynamic Drives, Solstice (Online) Disk Suite, and HP's Logical Volume Manager. How is yours configured? What devices are mirrored to what? How are your multidisk devices set up? Unbelievably, this information is not always captured by normal backup utilities. In fact, I used Logical Volume Manager for months before hearing about the `lvmcfgbackup` command (it backs up the LVM's configuration information). Sometimes if you have this properly documented, you may not need to restore at all. For example, if the operating system disk crashes, simply put the disks back the way they were and then rebuild the stripe in the same order, and the data should be intact. I've done this several times.

How are your databases set up?

> I have seen many database outages. When I ask a database administrator (DBA) how her database is set up, the answer is almost always, "I'm not sure…." Find out this information, and record it up front.

Did you document how you set up DHCP, Active Directory, NFS, and CIFS?

> Document, document, document! There are a hundred reasons to properly document things like this, and recovery from a disaster is one of them. Good documentation is definitely part of the backup plan. It should be regularly updated and available. No one should be standing around saying "I haven't set up NIS/AD/NFS from scratch in years. How do you do that again? Has anyone seen my copy of O'Reilly's book?" Actually, the best way to do this is to automate the

creation of new servers. If your operating system supports it, take the time to write scripts that automatically install various services, and configure them for your environment. Put these together in a toolkit that is run every time you create a new server. Better yet, see if your OS vendor has any products that automate new server installations, such as Sun's Jumpstart, HP's Ignite-UX, Linux Kickstart, and Mac OS cloning features.

Do you have a plan for this?

The reason for describing the earlier horrible scenarios is so you can start planning for them now. Don't wait until there's 20 feet of snow in your front yard before you start shopping for a snow shovel! It's going to snow; it's only a question of when. Take those pessimists out to lunch, let them dream of the worst things that could happen, and then plan for them. Have a fully documented, step-by-step plan for the end of the computer world as you know it. Even if the plan needs a little modification when you actually have to use it, you will be glad you have a starting point. It's a whole lot better than standing around saying, "What do we do now? Has anyone seen my résumé?" (You did keep a hardcopy of it, right?)

Know what's on your boxes!

The best insurance against almost any kind of loss is for the backup/recovery person to be familiar with the systems he is protecting. If a particular server goes down, you should know immediately that it contains an Oracle or SQL Server database and should be running for those volumes. That way, the moment the server is ready for a restore, so are you. Become very involved in the installation of any new system or database. You should know what database platforms you are using and

how they are set up. You should know about any new drives, filesystems, databases, or systems. You need to be very familiar with every box, what it does, and what's on it. This information is vital so that you can include any special backups for that type of system.

It Pays to Watch Your Logs

It was my very first gig out of college, so I was primarily supposed to be doing desktop support while learning at the feet of a high-priced Unix consultant, who we'll call *Fred*.

We were supporting a `ForEx` trading app called `Opus` that ran on SunOS. When it stored trades, half the information was in the path. For example, if someone made a USD-GBP trade on June 15 with someone from Bank of New York, the path and file would look like this:

```
/opt/app/opus/transactions/portfolio/third-
party/...etc...etc.../USD/CAI/GBP/BONY/ask/19970615120453.2372149821335
```

This insipid design was surely not Fred's fault, but he did set up the backups for it. I discovered that he had set up a `tar` job using `–v`, which produced logs so big he wasn't looking at them. Once I removed `–v` and started watching the logs, I found out that backups had been failing. The version of `tar` that shipped with SunOS at the time choked on file paths longer than 100 characters or so. The trading stubs were all about nine characters too long. Fred was basically tarring up a huge directory tree with no files at the bottom. Had he ever looked at the logs, he would have known that.

I was designated the primary Unix admin the next day. The company didn't renew Fred's contract.

Jim "Sparky" Donnellan

Are You Backing Up What You Think You're Backing Up?

I remember an administrator at one of my previous employers who used to say, "Are we getting this on tape?" He always said it with his trademark smirk, and it was his way of saying "Hi" to the backup guy. His question makes a point. There are some global ways that you can approach backups that may drastically improve their effectiveness. Before we examine whether to back up part or all of the system, let us examine the common practice of using include lists and why they are dangerous. Also, let's consider some of the ways that you can avoid using include lists.

What are include and exclude lists? Generically speaking, there are two ways to back up a system:

- You can tell your backup system to back up everything, except what is in an *exclude list*, for example:
- - For Unix, Linux, and Mac OS servers:
    - ```
      Include: *
      Exclude: /tmp, /junk1, /junk2
      ```
  - For Windows servers:
    - ```
      Include: *
      Exclude: *.tmp, *Temporary Internet Files*, ~*.*, *.mp3
      ```
- You can tell your backup system to back up what is in an *include list*, for example:
- - For Unix, Linux, and Mac OS servers:
    - ```
      Include: /data1, /data2, /data3
      ```
  - For Windows servers:
    - ```
      Include: D:\, E:\
      ```

Looking at these examples, ask yourself what happens when you create */data4* or the *F:* drive? Someone has to remember to add it to the include list, or it will not be backed up. This is a recipe for disaster. Unless you're the only one who adds drives or filesystems and you have perfect memory, there will always be a forgotten drive or filesystem. As long as there are other administrators and there is gray matter in your head, something will be left out.

I Hate It When That Happens

I was working at a major publishing company when an image server died. When those involved went to the backup administrator and asked for a restore of all the images from the server, he had no record of the server. It appears that after placing the server into production a year earlier, no one had formally requested that the server be added to the backup system. They lost thousands of images.

Chris Pritchard

However, unless your backup utility supports automated drive or filesystem discovery, it takes a little effort to say, "Back up everything." How do you make the list of what systems, drives, filesystems, and databases to back up? What you need to do is look at files such as */etc/vfstab* or the Windows registry and parse out a list of drives or filesystems to back up. You can then use exclude lists to exclude any drives or filesystems you don't want backed up.

Oracle has a similar file in Unix, called *oratab*, which can be used to list all Oracle instances on your server.[1] Windows stores this information in the registry, of course. You can use *oratab* to list all instances that need backing up.

Unfortunately, Informix and Sybase databases have no such file unless you manually make one. I do recommend making such a file for many reasons. It is much easier to standardize system startup and backups when you have such a file. If you design your startup scripts so that a database does not get started unless it is in this file, you can be reasonably sure that any databases that anyone cares about will be in this file. This means, of course, that any important databases are backed up without any manual intervention from you. It also means that you can use the same Informix and Sybase startup scripts on every system, instead of having to hardcode each database's name into the startup scripts.

How do you know what systems to back up? Although I never got around to it, one of the scripts I always wanted to write was a script that monitored the various host databases, looking for new systems. I wanted to get a complete list of all hosts from Domain Name System (DNS) and compare it against a master list. Once I found a new IP address, I would try to determine if the new IP address was alive. If it was alive, that would mean that there was a new host that possibly needed backing up. This would be an invaluable script; it would ensure there aren't any new systems on the network that the backups don't know about. Once you found a new IP address, you could use `nmap` to find out what type of system it is. `nmap` sends a malformed TCP packet to the IP address, and the address's response to that packet reveals which operating system it is based on.

Tip ✎

Some commercial data protection management software packages now support this functionality.

Back Up All or Part of the System?

Assuming you've covered things that are not covered by normal system backups, you are now in a position to decide whether you are going to back up your entire systems or just selected drives or filesystems from each system. These are definitely two different schools of thought. As far as I'm concerned, there are too many gotchas in the selected-filesystem option. Backing up everything is easier and safer than backing up from a list. You will find that most books stop right there and say "It's best to back up everything, but most people do something else." You will not see those words here. I think that not backing up everything is very dangerous. Consider the following comparison between the two methods.

Backing up only selected drives or filesystems

Here are the arguments for and against selective backups.

Save media space and network traffic.

The first argument that is typically stated as a plus to the selected-filesystem method is that you back up less data. People of this school recommend having two groups of backups: operating system data and regular data. The idea is that the operating system backups would be performed less often. Some would even recommend that they be performed only when you have a significant change, such as Windows security patches, an operating system upgrade, a patch installation, or a kernel rebuild. You would then back up your "regular" data daily.

The first problem with this argument is that it is outdated; just look at the size of the typical modern system. The operating system/data ratio is now significantly heavier on the data side. You won't be saving much space or network

traffic by not backing up the OS even on your full backups. When you consider incremental backups, the ratio gets even smaller. Operating system partitions have almost nothing of size that would be included in an incremental backup, unless it's something important that should be backed up! This includes Unix, Linux, and Mac OS files such as */etc/passwd*, */etc/hosts*, *syslog*, */var/adm/messages*, and any other files that would be helpful if you lost the operating system. It also includes the Windows registry. Filesystem swap is arguably the only completely worthless information that could be included on the OS disk, and it can be excluded with proper use of an exclude list.

Harder to administer.

Proponents of piecemeal backup would say that you can include important files such as the preceding ones in a special backup. The problem with that is it is so much more difficult than backing up everything. Assuming you exclude configuration files from most backups, you have to remember to do manual backups every time you change a configuration file or database. That means you have to do something *special* when you make a change. *Special is bad*. If you just back up everything, you can administer systems as you need to, without having to remember to back up before you change something.

Easier to split up between volumes.

One of the very few things that could be considered a plus is that if you split up your drives or filesystems into multiple backups, it is easier to split them between multiple volumes. If a backup of your system does not fit on one volume, it is easier to automate it by splitting it into two different include lists. However, in order to take advantage of this, you have to use include lists rather than exclude lists, and then you are subject to the limitations discussed

earlier. You should investigate whether your backup utility has a better way to solve this problem.

Easier to write a script to do it than to parse out the *fstab*, *oratab*, or Windows registry.

This one is hard to argue against. However, if you do take the time to do it right the first time, you never need to mess with include lists again. This reminds me of another favorite phrase of mine: “Never time to do it right, always time to do it over.” Take the time to do it right the first time.

The worst that happens? You overlook something!

In this scenario, the biggest benefits are that you save some time spent scripting up front, as well as a few bytes of network traffic. The worst possible side effect is that you overlook the drive or filesystem with your boss’s budget that just got deleted.

Backing up the entire system

The pros for backing up the entire system are briefer yet far more compelling:

Complete automation.

Once you go through the trouble of creating a script or program that works, you just need to monitor its logs. You can rest easy at night knowing that all your data is being backed up.

The worst that happens? You lose a friend in the network department.

You may increase your network traffic by a few percentage points, and the people looking after the wires might not like that. (That is, of course, until you restore the server where they keep their DNS source database.)

Backing up selected drives or filesystems is one of the most common mistakes that I find when evaluating a backup configuration. It is a very easy trap to fall into because of the time it saves you up front. Until you've been bitten though, you may not know how much danger you are in. If your backup setup uses include lists, I hope that this discussion convinces you to rethink that decision.

---

[1] You can install an Oracle instance without putting it in this file. However, that instance will not get started when the system reboots. This usually means that the DBA will take the time to put it in this file. More on that in Chapter 15.

Deciding When to Back Up

This might appear to be the most straightforward topic. Everybody backs up their system every night, right? What's the big deal? Actually, this could more aptly be titled "What levels do I run when?" It's always a big question. How often do you run a full backup? How often do you run incremental backups? Do you run various levels of incrementals that back up just today's changes or continuous incremental backups that back up everything since the last full backup? Everyone has her own answers to these questions. The only thing that is a definite is that there should be *at least* some level of backup every night. Before any further discussion on the topic, let's define some terms.

Backup Levels

The following are various backup levels. These terms are not used the same way by everyone.

Full/Level 0

: A full backup.

Level 1

: An incremental backup that backs up everything that has changed since the last level 0 backup. Repeated level 1 backups still back up everything since the last full/level 0 backup.

Levels 2–9

: Each level backs up whatever has changed since the last backup of the next-lowest level. That is, a level 2 backs up everything that changed since a level 1, or since a

level 0, if there is no level 1. With some products, repeated level 9 backups back up only things that have changed since the last level 9 backup, but this is far from universal.

Incremental

Usually, a backup backs up anything that has changed since the last backup of any type.

Differential

Most people refer to a differential as a backup that backs up everything that has changed since the last full backup, but this is not universal. In Windows, a differential is a backup that does not clear the archive bit. Therefore, if you run a full backup followed by several differential backups, they act like differential backups in the traditional sense. However, if you run even one incremental backup in Windows, it clears the archive bit, and the next differential backup backs up only those files that have changed since the last incremental backup. That's why a differential backup is not synonymous with a level 1 backup.

Cumulative incremental

I prefer this term to differential, and it refers to a backup that backs up all files that have changed since the last full backup.

Tip ✎

Backup products and backup administrators do not agree on these definitions. Make sure you know what your product means when it uses one of these terms!

The Windows Archive Bit Is Evil!

The Windows archive bit is evil and must be stopped. At the very least, backup vendors should give us the option of not using it—without penalty. If the “ready for archiving” bit is set on a file in Windows, it indicates that a file is new or changed, and that it should be backed up in an incremental backup. Once a file is backed up, the archive bit is cleared. Therefore, the first problem with the archive bit is that it should be called the *backup bit*; backups are not archives.

The biggest problem with the archive bit, however, is that the process assumes only one application will clear the archive bit, when there could actually be several of them. The first backup program to back up the directory clears the archive bit, and the next program does not back up the same files. Suppose a user decides to use `ntbackup` to back up to CD his files that are on the company’s file server. If he does that, `ntbackup` clears the archive bit, and the corporate backup system in charge of backing up those files will not back them up when it does an incremental backup. They don’t appear to be in need of backup because the archive bit is not set. This means that any user can defeat the purpose of the entire backup system.

Proponents of the archive bit point out that the archive bit is set on newly installed software, even if the files are old. A backup software package that uses only modification time does not “notice” these files if they’re older than the latest incremental backup, so perhaps what they should be using is a combination of the archive bit and modification time. If either has been changed, the file should be included in an incremental backup.

When backing up Unix systems, there is no archive bit, so backup applications use either `mtime` (when the contents of the file were last changed) or `ctime` (when the

attributes of the file were last changed). When backing up Windows systems, different backup applications use the archive bit differently. Some use it in conjunction with `mtime` and `ctime`. Some use only the archive bit, and others do not use it at all. (Based on what I'm saying about the archive bit, that might not be a bad thing.)

Microsoft has offered an alternative to the archive bit with the *change journal*, available in Windows 2000 and later. Backup products that support the change journal can consult it to determine which files have changed instead of looking at the archive bit. The change journal is not enabled by default, but it can be enabled using the `fsutil usn createjournal` command. You need to specify a `MaximumSize` that's big enough to hold all the changes that are made in between backups. Since 30 or 40 changes are stored in a single 4 K record, you can store 500,000 changes in a 75 MB journal. (If the change journal isn't large enough, the oldest changes are deleted from the beginning of the log to make room, so it's important to make the log large enough.) I suggest you find out the largest number of files you've ever had on an incremental backup and then make the log twice that size. The additional integrity this brings to your backup system more than makes up for the space this journal takes up.

A question that I am often asked is, "You want me to back up every night?" What the question really means is, "Even on the weekend?" Nobody's working on the weekend, right? *Right*…except for your noisiest customer last weekend. You know the customer I'm talking about: the one who calls your boss instead of the help desk when there's a problem. And if your boss isn't in or doesn't fix the problem fast enough, this customer will call your boss's boss. Well, last weekend this customer was really behind, so she spent

the entire weekend at work, working around the clock on next year's budget. She finally got it straightened out at about 1:00 a.m. Monday. At around 4:00 a.m., the disk where her home directory resides stopped working. (Everything dies Monday morning, doesn't it?) You haven't run a backup since Friday night. Your phone is ringing, and it's your boss. Any guesses as to what he wants to talk to you about? Do *you* want to be the one to tell this customer that you could have saved her file, but you don't run backups on the weekend?

Which Levels Do You Run and When?

There are several schools of thought on this question. The following are some suggested backup schedules.

Weekly schedule: All full/level 0 backups

Table 2-1 contains a backup schedule for the paranoid (not that paranoid is a bad thing). Performing a level 0 backup every day onto a separate volume. (Please don't overwrite yesterday's good level 0 backup with today's possibly corrupt level 0 backup!) If your system is really small, this schedule might work for you. If you have systems of any reasonable size, though, this schedule is not very scalable. It's also really not that necessary with today's commercial backup software systems.

Table 2-1. All full backups

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	Full/0	Full/0	Full/0	Full/0	Full/0	Full/0

Weekly schedule: Weekly full, daily level differentials/level 1s

The advantage to the schedule in Table 2-2 is that throughout most of the week, you would only need to restore from two volumes—the level 0 and the most recent level differential/level 1. This is because each differential/level 1 backs up all changes since the full backup on Sunday. Another advantage of this type of setup is that you get multiple copies of files that are changed early in the week. This is probably the best schedule to use if you are using simple utilities such as `dump`, `tar`, or `cpio` because they require you to do all the volume management. A two-volume restore is *much easier* than a six-volume restore—trust me!

Table 2-2. Weekly full backups, daily level differentials/level 1s

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	Diff/1	Diff/1	Diff/1	Diff/1	Diff/1	Diff/1

Weekly schedule: Weekly full, daily leveled backups

If your backup product supports multiple levels, you can use the schedule shown in Table 2-3. The advantage to this schedule is that it takes less time and uses less media than the preceding schedule. There are two disadvantages to this plan. First, each changed file gets backed up only once, which leaves you very susceptible to data loss if you have any media failures. Second, you would need six volumes to do a full restore on Friday. If you're using a good open-source backup utility or commercial backup utility, though the latter is really not a problem, because these utilities do all the volume management for you, including swapping tapes with an auto-changer.

Table 2-3. Weekly full backups, daily leveled backups

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	1	2	3	4	5	6

Weekly schedule: Monthly full, daily Tower of Hanoi incrementals

One of the most interesting ideas that I've seen is called the Tower of Hanoi (TOH) backup plan. It's based on an ancient mathematical progression puzzle by the same name. The game consists of three pegs and a number of different-sized rings inserted onto those pegs. A ring may not be placed on top of a ring with a smaller radius. The goal of the game is to move all of the rings from the first peg to the third peg, using the second peg for temporary storage when needed.[2]

A goal of most backup schedules is to put changed files on more than one volume while reducing total volume usage. The TOH accomplishes this better than any other schedule. If you use a TOH progression for your backup levels, most changed files are backed up twice—but only twice. Here are two different versions of the progression (they're related to the number of rings on the three pegs, by the way):

```
0 3 2 5 4 7 6 9 8 9
0 3 2 4 3 5 4 6 5 7 6 8 7 9 8
```

These mathematical progressions are actually pretty easy. Each consists of two interleaved series of numbers (e.g., 2 3 4 5 6 7 8 9 interleaved with 3 4 5 6 7 8 9). Table 2-4 uses a schedule to illustrate how this works.

Table 2-4. Basic Tower of Hanoi schedule

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
0	3	2	5	4	7	6

It starts with a level (full) on Sunday. Suppose that a file is changed on Monday. The level 3 on Monday would back up everything since the level 0, so that changed file would be included on Monday's backup. Suppose that on Tuesday we

change another file. Then on Tuesday night, the level 2 backup must look for a level that is lower, right? The level 3 on Monday is not lower, so it references the level 0 also. So the file that was changed on Monday, as well as the file that was changed on Tuesday, is backed up again. On Wednesday, the level 5 backs up only what changed that day, because it references the level 2 on Tuesday. But on Thursday, the level 4 does not reference the level 5 on Wednesday; it references the level 2 on Tuesday.

Note that the file that changed on Tuesday was backed up only once. To get around this problem, we use a modified TOH progression, dropping down to a level 1 backup each week, as shown in Table 2-5.

Table 2-5. Monthly Tower of Hanoi schedule

Day of the week	Week one	Week two	Week three	Week four
Sunday	0	1	1	1
Monday	3	3	3	3
Tuesday	2	2	2	2
Wednesday	5	5	5	5
Thursday	4	4	4	4
Friday	7	7	7	7
Saturday	6	6	6	6

If it doesn’t confuse you and your backup methodology,[3] and if your backup system supports it, I recommend the schedule depicted in Table 2-5. Each Sunday, you get a complete incremental backup of everything that has changed since the monthly full backup. During the rest of the week, every changed file is backed up twice—except for Wednesday’s files. This protects you from media failure better than any of the schedules mentioned previously. You will need more than one volume to do a full restore, of course, but this is not a problem if you have a sophisticated backup utility with volume management.

“In the Middle of the Night...”

This phrase from a Billy Joel song indicates the usual best time to do backups. Backups should be scheduled in such a way that they do not run during normal business hours. Sometimes you cannot avoid it, but it should not be a regular occurrence. There are two main reasons for this:

Integrity

> Unless you work in a 24/7 shop, nighttime is the time when the files are the most stable. (Of course, there could be batch jobs running that are manipulating data and customers accessing your web site, so not all files will be stable.) If you are backing up during the day, files are changing and probably also are open. Open files are more difficult to back up. Some backup packages handle open files better than others, but some cannot back them up at all. Also, if the file is changing throughout the day, you will not be sure what version you actually get on your backup.

Speed

> Another reason for not doing backups during the day is that the network is much busier, hence slower, during the day. The throughput of your backups slows significantly when your network is being used for normal traffic. If this is a problem at night as well, you might consider using a special network just for your backups. Doing backups during the day can significantly affect the speed of your other applications, and it is not good practice to regularly slow down your systems while people are using them.

Tip ✎

Of course, in today's global and Internet economy, "night" is relative. If you are in a shop in which the systems are accessed 24/7, you have to do things quite differently. You may want to look at Chapter 8 to see what vendors are doing to help meet this type of challenge.

It Does Have a Happy Ending (Almost)

Whew!

What's that? You think that I'm a mean and vicious person who is out to give you nightmares for the next week? You have no idea how you would get that information if you needed it? You say that you're going to lose sleep for a while? Good! Better to have lost sleep than to have lost data. One of the main purposes of this book is to scare you. A complacent person in charge of backups is a dangerous thing. The preceding scenario includes several Catch-22 situations and wipes out data that is not normally caught by standard backups.

---

[2] For a complete history of the game and a URL where you can play it on the Web, see http://www.math.toronto.edu/mathnet/games/towers.html.

[3] This is always the case for any recommendation in this book. If it confuses you or your backup methodology, it's not good! If your backups confuse you, you don't even want to try to restore! Always keep it simple, system administrator (K.I.S.S.).

Deciding How to Back Up

Once you've decided *when* you're going to back up, you have to decide *how* you are going to back up the data. But first, look at what types of problems you are protecting yourself from.

Be Ready for Anything: 10 Types of Disasters

As stated earlier, how you want to do your restores determines how you want to do your backups. One of the questions that you must ask yourself is, "What are you going to protect yourself from?" Are the users in your environment all "power users" who use their computers intelligently and never make dumb mistakes? Would your company lose a lot of essential data if the files on your users' PCs were accidentally deleted? If a hurricane took out your whole company, would it be able to continue doing business? Make sure that you are aware of all the potential causes for data loss, and then make sure your backup methods are prepared for all of them. The most exhaustive list of potential causes of data loss that I have seen is in another O'Reilly book called *Practical Unix and Internet Security* by Simson Garfinkel and Gene Spafford. Their list, with my comments attached, follows:

User error

> This has been, by far, the cause of the biggest percentage of restores in every environment that I have seen. "Hey, I was sklocking my flambality file, and I accidentally pressed the jankle button. Can you restore it, *please*?" This one is pretty easy, right? What about the common question: "Can you restore it as of about an

hour ago?" You can do this with continuous data protection systems and snapshots, but not if you're running backups once a night.

System-staff error

: This is less common than user error (unless your users have root or administrator privileges), but when it happens, oh boy, does it happen! What happens when you `newfs` your database's raw device or delete a user's document folder? These restores need to go *really fast*, because they're your fault. As far as protecting yourself from this type of error, the same is true here as for user errors: either typical nightly backups or snapshots can protect you from this .

Hardware failure

: Most books talk about protecting yourself from hardware failure, but they usually don't mention that hardware failure can come in two forms: disk drive failure and systemwide failure. It is important to mention this because it takes two entirely different methods to protect yourself from these failures. Many people do not take this into consideration when planning their data protection plan. For example, I have often heard the phrase, "I thought that disk was mirrored!" when a drive or filesystem is corrupted by a system panic. Mirroring does not protect you from a systemwide failure. As a friend used to say, if the loose electrons floating around your system decide to corrupt a drive or filesystem when your system goes down, "mirroring only makes the corruption more efficient." Neither do snapshots protect you from hardware failure—unless you have the snapshot on a backup volume.

Disk drive failure

Protecting your systems from disk drive failure is relatively simple now. Your only decision is how safe you want to be. Mirroring, often referred to as RAID 1, offers the best protection, but it doubles the cost of your initial drive and controller hardware investment. That is why most people choose one of the other levels of Redundant Arrays of Independent Disks (RAID), the most popular being RAID 5, with RAID 6 gaining ground. RAID 5 volumes protect against the loss of a single drive by calculating and storing parity information on each drive. RAID 6 adds more protection by storing parity twice, thus allowing for the failure of more than one drive.

Systemwide failure

Most of the protection against systemwide failure comes from good system administration procedures. Document your systems properly. Use your system logs and any other monitoring methods you have at your disposal to watch your systems closely. Respond to messages about bad disks, controllers, CPUs, and memory. Warnings about hardware failures are your chance to correct problems before they cause major disasters. Another method of protecting yourself is to use a journaling filesystem. Journaling treats the filesystem much like a database, keeping track of committed and partially committed writes to the filesystem. When a system is coming up, a journaling filesystem can roll back partially committed writes, thus “uncorrupting” the filesystem.

> **Tip ✎**
>
> The Windows change journal does not make NTFS a journaling filesystem in this sense. It contains only a list of files that have been changed; it does not actually contain the changes. Therefore, it cannot roll back any changes.

Software failure

: Protecting yourself from software failure can be difficult. Operating system bugs, database bugs, and system management software bugs can all cause data loss. Once again, the degree to which you protect yourself from these types of failures depends on which type of backups you use. Frequent snapshots or continuous data protection systems are the only way to truly protect against losing data, possibly a lot of data, from software failure.

Electronic break-ins, vandalism, and theft

: There have been numerous incidents of this in the past few years, and many have made national news. If you do lose data due to any one of these, it’s very different from other types of data loss. While you may recover the data, you can never be sure of what happened to the data while it wasn’t in your possession. Therefore, you need to do everything you can to ensure that this never happens. If you want to protect yourself from losing data in this manner, I highly recommend reading the book from which I borrowed this list, *Practical Unix and Internet Security*, by Simson Garfinkel and Gene Spafford (O’Reilly).

Natural disasters

Are you prepared for a hurricane, tornado, earthquake, or flood? If not, you're not alone. Imagine that your entire state was wiped out. If you are using off-site storage, is that facility close to you? Is it prepared to handle whatever type of natural disasters occur in your area? For example, if your office is in a flood zone, does your data storage company store your backups on the first floor? If they're in the flood zone as well, your data can be lost in one good rain. If you really want to ensure yourself against a major natural disaster, you should explore real-time, off-site storage at a remote location, discussed later in this chapter in the section "Off-Site Storage."

Other disasters

I remember how we used to test our disaster recovery plan at one company where I worked: we would pretend that some sort of truck blew up on the street that ran by our data center. The plan was to recover to an alternate building. This would mean that we would have to have off-site storage of media and an alternate site that was prepared to accommodate all our systems. A good way to do this is to separate your production and development systems and place them in different buildings. The development systems can then take the production systems' place if the production systems are damaged, or if power to the production building is interrupted.

Archival information

It is a terrible thing to realize that a rarely used but very important file is missing. It is even more terrible indeed to find out that it has been gone longer than your retention cycle. For example, you keep your backups for only three months, after which you reuse the oldest volume, overwriting any backups that are on that

volume. If that is the case, any files that have been missing for more than three months are *impossible* to recover. No matter how insistent the user is about how important the files are, no matter how many calls he makes to your supervisors, you will *never* be able to restore the files. That is why you should keep some of your backups a little bit longer. A normal practice is to set aside one full backup each month for a few years. If you're going to keep these backups for a long time, make sure you read the following sidebar "Are You Keeping Your Archives Too Long?" and the "Backup and Archive" section in Chapter 24.

"How Were the Backups Last Night?"

I suppose I've heard thousands of administrator-error horror stories, like people typing `rm -r /*` . I remember a guy who wanted to delete a junk file in */bin* called *?*&(&^JI($SF))FS%$#T*, or something like that. He typed `rm /bin/?*` (which deleted all the files starting with any character—that's right—all of them). But there's one story that I witnessed firsthand that still makes me laugh.

A consultant was given the task of cleaning up our home directories. Apparently, my company was very good about deleting logins for people who had left the company, but we weren't very good about deleting their home directories. The consultant wrote a program that basically did the following:

`cd` into */home1*

`find`, looking for directories that did not match an entry in the password file and were not owned by root or administrator

`rm -r` that directory

Each user's home directory was located under a directory that was the first letter of her login. For example, the home directory for *cpreston* was in */home1/c/cpreston*. The scenario went something like this. The idea was that */home1/c* would be owned by root and thus would not be deleted. Unfortunately, over the years, an administrator or two would `cd` into */home1/c/cpreston* and try to correct an ownership problem. To do that, the administrator would type `chown cpreston .*`. Well, if you've ever done that as root, you know that **.*** includes **..**, which in this case would be */home1/c*. Thus, the */home1/ c* ends up being owned by me!

The consultant did not foresee this and so would interpret */home1/c* as a user's home directory and look for the user called "c" in the password file. Of course, there was no such user, so the program said `rm -r /home1/c`. I'm not sure when my friend realized what was happening, but I do remember being on my way out the door and getting a weird phone call. "How were the backups of */home1* last night?" my friend asked—very sheepishly and very mysteriously. "Fine, as always," was my response, "Why?" There's something beautiful about the power that the backup guy yields at that magic moment when someone *really* needs some files restored. Up to that point, you're the guy who comes in early and stays late, watching the backup drives spin. In one moment, you're transformed into the most important person he knows! *Cool*.

Automate Your Backup

If you work in a shop with a modest budget, you probably looked at this heading and said, "Sure, if I could afford it." Although automation that involves expensive jukeboxes and autochangers is nice, that is not the type of automation I

am talking about. There are two types of automation. One type allows your backups to complete an entire cycle without requiring any manual intervention from you, such as ejecting and loading new volumes. This type of automation can make things much easier but can also make them much more expensive. If you can't afford it, a less expensive alternative is to have your backup system notify you when you need to do something manually. At the very least, it should notify you if you need (or forgot) to change a volume. If things aren't going right, you need to know. Too many times people look at their backup logs only when they need to do a restore. That's when they find out that their backups have failed for days or weeks. A slightly intelligent backup system could email you or page you if things don't go the way you expect them to go.

The second type of automation is actually much more important. This type of automation refers to how your backups "think." Your backup process should know what to back up without you telling it. If a DBA installs a new database, your backups should know about it. If a system administrator installs a new drive or filesystem, your backups should automatically include it. This is the type of automation that is essential to safe backups. A good backup system should not depend on a human brain to remember to do something.

Are You Keeping Your Archives Too Long?

Some governments have laws and regulations that govern how long certain types of data are allowed to be kept in a company's files. We're not talking about regulations that say you must keep data for a certain number of years. We're talking about a regulation that says you must *delete* data after a certain number of years. For example, you may be told that your personnel department can keep disciplinary paperwork for only two years. If an employee

believes that her chances for a promotion are reduced because of a disciplinary action that is more than two years old, she can sue for damages. Many lawsuits have been filed based on these laws.

What happens when the disciplinary action “paperwork” is actually a file on someone’s computer? The laws extend to the computers too, and the files must be deleted. But what if that file is on an archive volume that is being kept forever? Many companies have backup policies that dictate that one volume per system per year is kept “forever.” In recent years, some companies have lost lawsuits because of policies like this.

The only way around this is to exclude from regular backups any directories that contain this type of information and archive them using a different schedule that conforms to the document retention laws of your state. I admit this is a pain. You will never read that I think that doing something special for anything is a good thing, but in these litigious times, this issue should not be overlooked.

Plan for Expansion

Another common problem happens as a backup system grows over time. What works for one or two boxes doesn’t necessarily work for 200. As the volume of data grows, the need for a standardized backup system becomes greater and greater. This is a problem because most administrators, as they are writing their shell script to back up five or six boxes, do not think ahead to the time when there may be many more. I can remember my early days as the backup guy. I had 10 or 11 systems, and the “monster” was an Ultrix box. It was “huge,” we said in those days. (It was almost 8 gigabytes!) The smallest tape drive we had

was a 10 GB (with compression) Exabyte. We used the big 10 GB tape drive for the 8 GB system. We had what I considered to be a pretty good in-house backup script that worked without modification for two years.

Then came the HPs. The *smallest* system was 20 GB, and the biggest was much bigger than that. But these big systems came with a little 2 GB (4 with compression) DDS drive. Our backup script author never dreamed of a system that was bigger than a tape. One day I woke up, and our system was broken. I then spent months and months hacking up that shell script to support splitting the drive or filesystem into two tapes. Eventually, I gave up and bought a commercial product. My point is that if I had thought of that ahead of time, I might have been able to overcome the limitation without losing so much sleep.

When you are designing your backup system—or your data center, for that matter—plan on your systems getting bigger and more numerous. Plan for what you will do when that happens—trust me, it *will* happen. It will be much better for your mental health (not to mention your job security) if you can foresee the inevitable and plan for it when you design the system the first time. Your backup system is something that should be done right the first time. And if you spend a little time dreaming about how to break it *before* you design it, you can save yourself a lot of money in antacids and sleeping pills.

Don’t Forget Unix mtime, atime, and ctime

Unix, Linux, and Mac OS systems record three different times for each file. The first is `mtime` , or modification time. The `mtime` value is changed whenever the contents of the file have changed, such as when you add lines to a logfile. The second is `atime` , or access time. The `atime` value is

changed whenever the file is accessed, such as when a script is run or a document is read. The last is `ctime` , or change time. The `ctime` value is updated whenever the attributes of the file, such as its permissions or ownership, are changed.

Administrators use `ctime` to look for hackers because they may change permissions of a file to try to exploit your system. Administrators also monitor `atime` to look for large files that have not been accessed for a long time. (Such files can be archived and deleted.)

Backups change atime

You may be wondering what this has to do with backups. You need to understand that any backup utility that backs up using the filesystem modifies `atime` as it reads the file to back it up. Almost all commercial utilities, as well as `tar` , `cpio`, and `dd`,[4] have this feature. `dump` reads the filesystem via the raw device, so it does not change `atime`.

The atime can be reset—with a penalty

A backup program can look at a file's `atime` before it backs it up. After it backs up the file, the `atime` obviously has changed. It can then use the `utime` system call to reset `atime` to its original value. However, changing `atime` is considered an attribute change, which means that it changes `ctime`. This means that when you use a utility such as `cpio` or `gtar` that can reset `atime`, you change `ctime` on every file that it backs up. If you have a system that is watching for `ctime` changes, it will think that it's found a hacker for sure!

Make sure that you understand how your utility handles this issue.

Don’t Forget ACLs

Windows files stored on an NTFS filesystem and some files stored on modern Linux filesystems use access control lists (ACLs) to grant or restrict permissions to users. ACLs say who can read, write, execute, modify, or have full control over a file. Figure 2-1 shows an example of such ACLs.

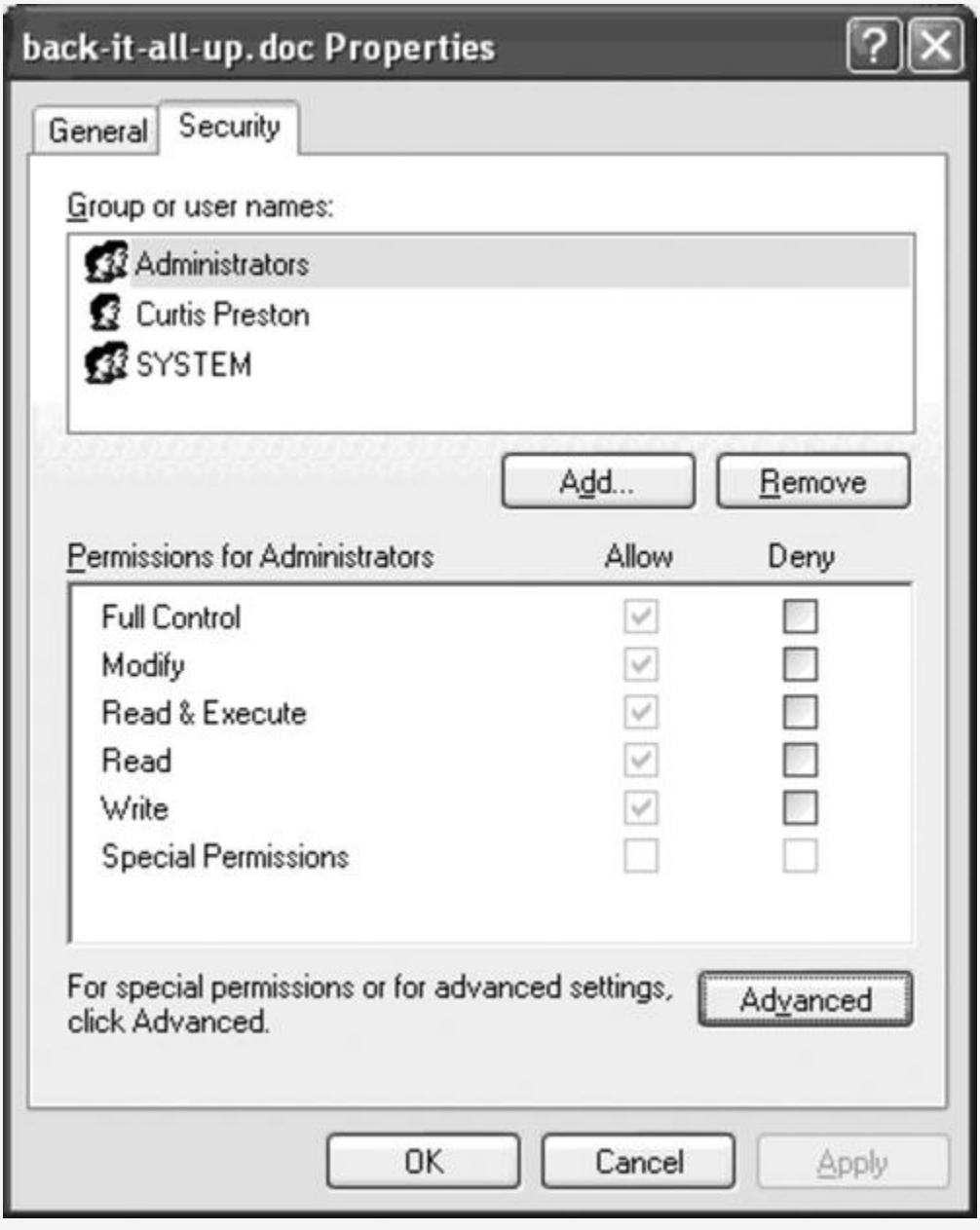


Figure 2-1. Access control list example

You need to investigate how your backup product is handling ACLs. The proper answer is that they are backed up and restored. This is a feature common with commercial products, but unfortunately, not all open-source products do this. Make sure you look into this when evaluating open-source tools.

Don’t Forget Mac OS Resource Forks

Mac OS files stored in MFS, HFS, or HFS Plus filesystems have two forks: the data fork and the resource fork. The data fork contains the actual data for the file, such as its text. The resource fork contains related structured data, such as offsets, menus, dialog boxes, and icons for the file. The two forks are tightly bound into a single file. While they are typically used by executables, every file can have a resource fork, and other applications can use it as well. For example, a word processing program may store a file’s text in the data fork and the file’s images in the file’s resource fork.

These resource forks, like Windows ACLs, need to be backed up, and not all backup products back them up properly. Make sure you investigate what your backup system does with the data fork and resource fork.

Keep It Simple, SA

K.I.S.S. Have you seen this acronym before? It applies double or triple to backups. The more complicated your backup scheme is, the more likely it is to fail. If you do not understand it, you cannot implement it. Remember this every time you consider adding a new bell or whistle to your backup system. Every change puts your data at risk. Also, every change might make your backup system that

much more complex—and more difficult to explain to the new backup person. One of the heads of support for a commercial backup product said that he sees the same thing over and over again. One person gets to know the software really well and writes various scripts to automate this and that. Backups become a well-oiled machine—until they are turned over to the trainee. The trainee doesn't understand all the bells and whistles, and things start breaking. All of a sudden, your data is in danger. Keep that in mind the next time you think about adding some cool new feature to your backup script.

This next comment also relates to the previous section about "thinking big." One of the common judgment errors is to not automate in the beginning. It's so much easier to just put a hardcoded include list in a file somewhere or put it in the `cron` or scheduled task entry itself. However, that creates many different backup methods. If each box has its own special customized backup system, it is very hard to monitor your backups and explain them to the new person.

> **Tip ✎**
>
> Remember, special is bad. Just keep saying it over and over again until you believe it.

It's not such a big deal when you have two or three systems, but it is when you grow to 200 systems. If you have to remember every system's idiosyncrasies every time you look at your logs, things inevitably get out of control. Exceptions for each system also can mean that things get overlooked. Do you remember that nine months ago you excluded */home** on *apollo*? I hope so, if *apollo* just became your primary NFS server, and it now has seven home directories.

If you cannot explain your backups to a stranger in less than a few hours, things are probably too complex. You should look at implementing things like centralized logging, standardized backup scripts, and some level of automation.

Read Those Manuals

The IP address of the backup server for a large software company was constantly changing to different, seemingly random IP addresses. The only identifiable pattern was that each new IP address the backup server would be assigned would be an IP address of one of the backup clients. Support cases were opened with vendors and all engineers were working 24/7 to resolve it, yet nobody could figure it out.

It turned out that a backup operator assigned to resolving backup issues was troubleshooting using the standard troubleshooting procedures for the group. But the new backup operator mixed up a few commands, so when trying to do basic name resolutions for the backup hosts (`nslookup` *`hostname`*), the command issued became `ifconfig -a` *`hostname`* instead. This changed the IP address of the backup server to whatever host was having backup issues, at random times of the day, and only on the days that operator was working.

Jorgen Lie

---

[4] `dd` has this feature when you're using it to copy an individual file in a filesystem, of course. When using `dd` to copy a raw device, you will not change the access times of files in the filesystem.

Storing Your Backups

It doesn’t do any good to make really good backups only to have your backup volumes destroyed, lost, or misplaced. You need to have a well-defined process for storing your media.

Storage in General

If you’ve read this far, you know that I consider your backups very important. If your backups are important, isn’t the media on which they reside just as important? That goes without saying, right? Well, you’d never know it from most volume “libraries.” Volume “piles” is probably a more accurate term. How many computer rooms have you seen that have volumes spread out all over the place? They get stacked, piled, fall behind the systems, and a tape cartridge works really well as a coaster for a coffee mug. (We wouldn’t want to get any coffee rings on the new server, right?)

Have you ever really needed a volume and couldn’t find it? I’ve been there. It’s a horrible feeling to know that you’ve got the file on a volume, but can’t find the darn volume! Why, then, do we treat our backup volumes like so much dirty laundry? Organize your backup volumes! Label them, catalog them, give them unique names or numbers, and put them in some sort of logical order in some kind of storage container. Do it, or the backup demon will come to haunt you!

Tip ✎

Your ability to perform a large recovery quickly is directly related to how well you organize your media.

On-Site Storage

What about that media cabinet that you're using for your on-site volume storage? You don't have one, you say? You're using a file cabinet, you say? Well, use something, but if you can afford it, a number of companies make storage containers for media. They also make cabinets that can withstand fire. Spend the money; you'll be glad you did. Doing a restore is so much less stressful when you can find the volume with no problem. Remember, though, that fireproof does not mean heat-proof. These types of media safes are meant to withstand brief fires that are quickly extinguished by a sprinkler system. If a fire burns for a long time right next to the container or raises the temperature in the room significantly, the volumes may be no good anyway. (This is another good reason why you also must store volumes off-site.)

Have the most well-organized person in your office design your media storage system. Here's an idea. Ask your best administrative person to take a look at your storage system and compare it to his filing cabinet. Explain that you want an honest evaluation.

12,000 gold pieces

A financial institution where I once worked had an inventory of more than 12,000 pieces of media, and we never lost one. How did we do it, you ask? We treated every

volume as if it were a piece of gold. Our inventory system was built on a number of things:

- Each volume had a unique numeric identifier.
- This number was in the form of a bar code placed on *every volume*. (Labeling more than 500 $5^1/_4$-inch original installation floppies that came with our AT&T 3b2/1000s was no joy, I assure you, but we did it with the help of a team of temps!)
- Each volume's number, name, purpose, media type, date used, and location were stored in an Informix database.
- Every volume movement was tracked by that database. When a volume was taken to another building for a backup or restore, that movement was recorded in the database. If a volume was sent to our off-site storage vendor, that was stored in the database. If an administrator borrowed a backup volume or installation CD, it was recorded in a field called "Loaned to:".
- There was a manual log for when we moved media out of the media library momentarily for restores. For daily, high-volume moves, we used a bar code scanner with a shell script that automatically updated the database.
- We did a complete inventory every other quarter and a spot-check inventory once a month. If the spot-check inventory turned up too many errors, it was time for another full inventory.
- During the inventory we checked every volume against a printout of the database and every entry in the printout against an actual volume. (The latter half of the inventory consisted of hunting down errant administrators who had squirreled away backups or installation media in their drawers.)

- The volumes were stored in Wrightline media cabinets and were behind locked doors. Only the backup operators had access to the volumes. (These were the same operators who were held responsible if something came up missing.)
- The inventories were called “self-audits,” and there was also an annual internal audit by the audit department, as well as the external audit by the Office of the Comptroller of Currency. They would comb through our logs, looking for inconsistencies. They had a knack for finding entries that looked a little weird and saying, “Let me see this one….”
- This entire process was thoroughly documented, and the institution is still following these procedures to this day, although it has probably improved them a bit.

The OCC takes this whole issue of data protection very seriously. (The OCC, by the way, is the group that has the power to say you can no longer be a bank. You want to make sure that they are happy with your procedures.)

Off-Site Storage

Once you have organized the media that you are storing on-site, it’s time to consider off-site storage. There are two ways to store your data off-site:

- Media vaulting (they hold your tapes)
- Electronic vaulting (no tapes)

The latter can be expensive, but not as expensive as some people think. It is also much easier to use during a disaster, and you can’t lose tapes if there aren’t any tapes to lose. That is, of course, what off-site storage is meant to prepare

you for—the destruction of your media and/or the building that holds it. If you have a complete set of backups in another location, you will be able to recover from even the worst local disaster.

Choosing a media vaulting vendor

Choosing a media vaulting vendor is as important a task as choosing your backup software. Choosing the wrong vendor can be disastrous. You depend on that vendor as your last line of defense, which is why you are paying them. Therefore, their storage and filing procedures need to be above reproach. They need to be better than the scenario I described in the “12,000 gold pieces” section earlier in this chapter. Their movement-tracking procedure must be free of holes. Here is a list of things to consider when choosing an off-site storage vendor:

Individual media accountability

: The first media vaulting vendor I ever used stored all of my volumes inside cases. They never inventoried the individual pieces of media. It was up to me to know which volume was in which case. When I needed a volume from one of the cases, they had to go in and get it. Once that was done, there was no log of where that volume actually existed. This is referred to as *container vaulting*. Most media vaulting companies also offer individual media vaulting. This method ensures that every volume is being tracked.

Bar-coded, location-based inventory

: Again, each volume should have a bar code that allows your storage vendor to scan every volume in and out. They should scan volumes into their vault when they

arrive and scan them out when they give them back to you.

Electronic double check

If you are keeping track of every volume's location, and your vendor is too, you should double-check each other. One or both of you can print out an export of your database that shows volume locations. You can write a program that cross-checks the location of every volume against the other inventory. I can't tell you how many times such a program has saved me. It's great to find an error when it happens, instead of weeks later when you need a volume that got misplaced.

Testing your chosen vendor

See if your vendor is on their toes. One tricky thing you can do is to see if they leave you alone in the vault. You are a customer of this company, so ask them if you can do an inventory of your media alone. See if they allow you unrestricted access to the inside of the vault. If they leave you alone inside the vault with no supervision, you have access to other companies' media. That means that at certain times, other companies may have access to your media. Run, don't walk, away from this company.

Make surprise inspections. Make spot checks. Ask for random volumes back, and see how quickly they can find them. Ask for volumes you just sent them. Volumes in the process of being inventoried are the hardest to find, but they should be able to do it. If you regularly send them five volumes a day with an inventory, put four volumes in one day, but list five on the inventory. See if they notice. If they don't, raise a ruckus! Their procedures should protect you from these types of human errors. If they don't, those procedures need to be improved. Be unpredictable. If you

become predictable, you may be overlooked. Keeping them on their toes will make them remember you—and how important you think your volumes are. (By the way, your ability to make surprise inspections and spot checks should be spelled out in your contract. Make sure that it is OK for you to do this. If it is not…well, you know what to do.)

Vendors store two types of volumes: those that rotate in and out and those that stay there indefinitely. As you rotate the cyclical volumes in and out, they are inventoried. Your archive volumes are another story. If a volume has been there for two years and has never been touched, how do you know that it's OK? You should make a full inventory of those volumes at least once, preferably twice, every year.

Tip ✎

Send the original, keep the copies. One of the things that you should regularly test is your copy procedure. If you are sending volumes off-site, some backup products give you the option of sending the originals or copies. If you can, send the originals. When it comes time for a restore, use your copy. If things go wrong, you can always go get the original. This process validates your copy procedure every time you do a restore. You can correct flaws in the process before disaster strikes. I remember several instances when a volume was eaten in a drive, or had soda spilled on it, and we needed that off-site copy really badly. That is the wrong time to find out your copy procedure is no good!

Electronic vaulting

Electronic vaulting is becoming quite popular. It can be expensive, but it's a beautiful thing. If you can afford it, I

highly recommend it. The premise is that your backups are sent directly to a storage system at the electronic vaulting vendor. One question you need to ask yourself is, “What happens if *they* burn to the ground?” All your data could be lost. Don’t let this happen. Make sure that this storage company is not the only location for your backed-up data. In addition, make sure that you know how you’re going to do a large restore. While a small network link may be large enough to do a continuous incremental backup, it’s probably not large enough to do a 100 GB restore. If this is a concern, ask your electronic vaulting vendor about a local recovery appliance.

Testing Your Backups

I wish there were enough to say about this to make it a separate chapter, because it's that important. I can't tell you how many stories I have heard about people who waited until they needed a major restore before they tested their backups. That's when they found out that they'd been using the wrong device or the wrong blocking factor, or that the device had I/O errors. This point cannot be stated strongly enough. If you don't test your backups, you are guaranteed to get a surprise sooner or later.

Test Everything!

It is important to test every type of restore. If you are testing filesystem backups, make sure you:

- Restore many single files. Can you find the needle in the haystack?
- Restore an older version of a file.
- Restore an entire drive or filesystem, and compare your results with the original. Are they the same size, and so on?
- Pretend that an entire system is down, and try to recreate it.
- Pretend that a particular volume is bad, and force yourself to use an alternate backup.
- Retrieve a few volumes from your off-site storage vendor.
- Pretend that your backup server is destroyed, and try to recover from that. (This one's tough!) This test is extremely important if you are using an open-source or

commercial backup utility. Some products do not plan for this well, and you can find yourself in a real Catch-22 situation.

If you are testing database restores, make sure you:

- Restore part of your database, pretending that you lost only one data file or disk drive, if this option is available.
- Restore the entire database onto another server; this is where you learn about files that you are not including.
- Restore the database up to a point in time, earlier than the present time (this is helpful practice for recovering from a DBA or user error).
- Pretend that last night's backup failed, and force yourself to use an older backup. Theoretically, if you have saved all your transaction logs to a backup volume, you should be able to use a backup that is weeks old and roll it forward to the present time using those logs. This is another strong argument for using transaction logs.

Test Often

As I said earlier, sit around one day with some really pessimistic people, and ask them to dream up scenarios for you to test. Test your ability to recover from each of these scenarios *on a regular basis*. What works this month might not work next month. The only thing that is guaranteed to remain constant is change. One suggestion is to create a list of recovery procedures and randomly test a subset of them every month. Management changes, hardware changes, networks change, and OS and database versions

change. Every change you know about should make you want to perform a test of the affected area.

Monitoring Your Backups

If you are not monitoring your backups, they are not doing what you think they are doing—guaranteed. This is one pot that will not boil if you don't watch it. Every backup should have a log that is examined daily. This can be automated as well. Here are some examples:

Give me a summary.

`dump` gives a whole bunch of messages that I couldn't care less about, `Pass I`, `Pass II`, `% done`, and so on. When I'm monitoring the `dump` backups of hundreds of drives or filesystems, most of that is so much noise. What I really want to see is what got dumped, where it went, when it went, what level it was, and the ever-popular `DUMP IS DONE` message. To get a summary of just these lines, the first thing I do is use `grep -v` to exclude the phrases I don't want, leaving only a few lines. This is much easier to review. This technique can also be applied to other Unix, Linux, and Mac OS backup commands.

Show me anything weird.

You can do this in either of two ways. If you know the phrases that show up when things go wrong, `grep` for those. Another way is to use `grep -v` to remove all lines you're expecting and see what's left. If there's nothing, great! If there are lines left over, they are probably errors. You may see lines such as `I/O error`, `Write error`, or something else you don't like to see in your backups.

If you want to apply this to a Windows task, you need to take advantage of some Unix emulators, such as Cygwin, UWIN, or GnuWin32, to allow you to run `grep` and other shell commands on a Windows system.

You Can Always Make It Better

I don’t care how good your backups are; they can always be better. You could spend every waking hour tweaking and improving every piece of your backup program and know everything there is to know about backups, and they could still be better. My backups will never be good enough. There’s always a new bell or whistle on some other backup package, a bigger or smarter jukebox, a faster backup drive, or some scenario I thought of that I’m not covering. You must realize, however, that every change you make has a potential for data loss. A common thread that you will find in this book is that every time the human being enters into the equation, things can go wrong. You may be the best shell or Perl hacker in the world, and you will still make mistakes.

If It’s Not Baroque, Don’t Fix It

Baroque needed fixing. Come on! One constant tempo? And the harpsichord? It had to go. Thankfully, Bartolomeo Cristofori invented the piano in 1709 and gave us an instrument that could play loud (fortissimo) and quiet (piano)—hence its name. Shortly after that, the Classical period started, and music started changing in tempo and feeling.

But if it’s *not* broken, don’t fix it! You’ve heard it before, but given the risk that each change makes, it goes double in the backup world. As you read this book or some magazine, or talk to other administrators, you will undoubtedly come up with a list of things that you wish you were doing. Concentrate on the *holes*, or scenarios that your backup and recovery plan just does not cover. Worry about the fact that none of your volumes are stored off-site before you think

about working on that cool menu program you've been wanting to write for your restores. Make sure you're covering all the bases before you start redecorating the stands. Before you consider making a new change, ask yourself if something else is more important and if the change is really necessary and worth the risk.

Following Proper Development Procedures

Don’t make a new change on your backup system and then roll it out to all your machines at once. Test it on a development system, or better yet, on a system that you don’t normally back up. That way you aren’t putting any backups in jeopardy. Another good practice is to test the change in parallel with what you’re already doing. The bigger the change, the more important it is to do a parallel conversion. This is especially true if you’re using a new method rather than just enhancing your current one. Don’t stop using your old method until you’re sure that the new one works! Follow a plan similar to this:

- Test backup changes somewhere where they really won’t hurt anybody if they do something like, oh, crash the system!
- Test the operation on a small scale on one system, using it in the same manner as you would in production. For example, if you are going to do both remote and local backups with this program, test both on a small scale.
- Try to simulate every potential error the system might encounter:
-
    - Eject a volume in the middle of the backup.
    - Write-protect a volume.
    - Reboot the system you are backing up while it is backing up.
    - Drop the network connection, and power down a disk drive.
    - Know the system and the errors for which it is testing, and simulate each one to test that section

of your system.

- Test on a small number of systems, preferably in parallel with your current method.
- When you roll it out to all systems, definitely do so in parallel. One of the ways you can do this is to squeeze all your backups onto as few volumes as you can, then use the leftover drives to do the new backup in parallel. Your network guys might hate you, but it's really the only way to do a true parallel conversion. When I converted to my first commercial backup utility, I ran in this mode for almost a year.
- Only after you've tested and thoroughly documented your new system should you turn off the old method. Remember to keep documentation and programs around to restore data from the old system until all the old volumes have been recycled into the new system.
- Also consider your older backup volumes. If you have volumes that are five years old, are you going to be able to read them on a new vendor's backup solution? Will you even be able to read them in version 14 of your current software if your company began writing the archive volumes in version 2? Will the media itself even be readable?
- This gets into a whole other subject: media life. Even if you could still theoretically read the volumes 12 versions later, the volumes are probably bad. If you have long-term archives, you need to make fresh copies of them at some set interval by copying older tapes to newer tapes.

Unrelated Miscellanea

We were going to call this section “Oh, and by the way,” but that seemed like a really weird heading.

Protect Your Career

One of the reasons that backups are unpopular is that people are worried that they might get fired if they do them wrong. People do get in trouble when restores don’t go right, but following the suggestions in this section will help you protect yourself from “recovery failure fallout.”

Self-preservation: Document, document, document

Have you ever tried to go on vacation? If you’re the only one who understands the restore process or the organization of your media, you can bet that you will be called if a big restore is required. Backups are one area of system administration in which inadequate documentation can really get you in trouble. It’s hard to go on vacation, get promoted, or do anything that would pull you away from the magical area that only you know. Your backups and restores should be documented to the point that any system administrator can follow them step by step in your absence. That is actually a good way to test your documentation: have someone else try to use it.

The opposite of good documentation is, of course, bad or nonexistent documentation. Bad documentation is the surest way to help you find a new job. If you do ever manage to take a real vacation in which you don’t carry a beeper, check your voice mail, or check your email—in short, watch out: Murphy’s Law governs vacations as well. You can guarantee that you, or more accurately, your

coworkers, will have a major outage that week. If they crash and burn because you left them no guidelines for how to perform a restore, they will be looking for you when you return. You will not be a popular person, and you just might find yourself combing through the want ads.

You Can Run, But You Can't Hide

On a number of occassions, a lack of documentation caused me to lose personal time. I remember one vacation during which I spent two to three hours on the phone every day. I remember spending long nights in computer rooms because no one knew which button to press next. But none of those memories is as strong as the time when my wonderful daughter, Nina, was born. Right about now, you're probably saying "Aaah, that's sweet." It's not what you think. Yes, she and my other daughter, Marissa, have given me a whole other reason to get up every day, but that's not what this story's about.

The hospital in which my wife gave birth was about two blocks from my office building. I knew that. My coworkers knew that. (Anybody who looked out the window knew that!) The day Nina was born, we lost a major filesystem. I knew it was on a backup volume, and I knew I was *off duty*. I left my beeper, which is normally welded to my side, at home. I did not call in to work. I knew the process was documented. The problem was that they weren't reading the documentation. "Call Curtis!" I was standing in my wife's hospital room, talking about our wonderful child, when the phone rang. Those guys tracked me down and called me in the hospital! They asked me to come in, but because I knew the system was documented well, the answer was *no!* (I think I actually hung up without saying another word.) This is an example of the lengths to which people will go to find you if you

don't have proper documentation or if they have not been shown how to use it.

Documentation is also an important method of letting your internal customers know what you are doing. For example, if you skip certain types of files, drives, or filesystems, it is good if you let people know that. I remember at least one very long conversation with a user who really didn't want to hear that I didn't back up */tmp:* "I never knew that *tmp* was short for temporary!"

Strategy: Make backups an integral part of the installation process

When a new system comes in the door, someone makes sure that it has power. Someone is responsible for the network connection, assigning an IP address, adding it to the NIS configuration, and installing the appropriate patches. All those things happen because things don't work if they don't happen. Unfortunately, no one notices if you don't add the machine to your backup list. That is, of course, until it crashes, and they need something restored. You have the difficult task of making something as "unimportant" as backups become just as natural as adding the network connection.

Tip ✎

A new system coming in the door is usually the best test machine for a complete server recovery/duplication test. Not many people miss a machine they don't have yet.

The only way that this is going to happen is if you become very involved in the whole process. Perhaps you are a

junior person, and you never sit in on the planning meetings because you don't understand what's going on. Perhaps you *do* understand what's going on but just *hate* to go to meetings. So do I. If you don't want to attend every meeting, just make sure that someone is looking out for your interests in those meetings. Maybe an ex-backup operator who goes to the meetings and is sympathetic to your cause. Have briefings with her, and remind her to make sure that backup needs are being addressed or to let you know about any new systems that are coming down the pike. Occasionally, go to the meetings yourself, and make sure that people know that you and your backups exist; hopefully, they'll remember that the next time they think about installing a new system without telling you. Never count on this happening, though. You've got to be ever-diligent, looking for new systems in need of backup.

New installations are not the only thing that can affect your backups. New versions of the operating system, new patches, and new database versions all can break your backups. Most system administrators bring in a new version of their operating system or database and run it on a new box or development box before they commit it to production. Make sure that your backup programs run on the new platform as well. I can think of a number of times that new versions broke my backups. Here are a few examples:

- When HP-UX 10 came out, it supported file sizes greater than 2 GB, but the `dump` manpage said that `dump` does not back up a filesystem with large files.
- Oracle has changed a number of times. Sometimes it maintains backward compatibility; sometimes it doesn't.

- The Windows Encrypting File System couldn't be backed up by some backup systems when it first came out.
- Mac OS versions (post Mac OS X) always seem to be a little ahead of the backup curve. The methods you used to back up or image the last version no longer work in the new version, giving rise to many unofficial versions of tools that actually work.

The longer I think about it, the more stories I come up with. If you've been doing this for a while, I'm sure you have a few of your own. Suffice it to say that OS and application upgrades can and do cause problems for the backup person. Test, test, test!

Get the Money Your Backups Need

This final section has absolutely nothing to do with backups. It has to do with politics, budgeting, money, and cost justifications. I know that sometimes it sounds as if I think that backups get no respect. Maybe you work at Utopia, Inc., where the first thing they think about is backups. The rest of us, on the other hand, have to fight for every volume, drive, and piece of software that we buy in order to accomplish this increasingly difficult task of getting it all on a backup volume.

Getting the money you need to accomplish your task can sometimes be very difficult. Once a million dollar computer is rolled in the door and uncrated, how do you tell the appropriate department that the small, standalone backup drive that came with it just isn't going to cut it? Do you know how many hoops they went through to spend a million dollars on one machine? You want them to spend how much more?

Be ready

The first thing is to be prepared. Be ready to justify what you need. Be ready with information such as:

- Statistics on recoveries that you have performed.
- Any numbers that you have on what downtime and lost data would cost the company, including any recovery-time-objective and recovery-point-objective requirements from business units (RTO and RPO are covered in more detail in Chapter 24).
- Numbers that demonstrate how a purchase would help reduce staff costs.
- Numbers that demonstrate how the current backup system is being negatively impacted by growth or new applications.
- Cost comparisons between the one-time cost of a newer storage system and the continuing cost of the manual labor required to swap volumes every night (be prepared to explain how a larger jukebox or virtual tape library reduces the chance for human error and how that helps the company as well).
- A documented policy that every new gigabyte results in a surcharge of a certain amount of money.

A well-designed presentation of what service your backups will provide and the speed at which you can recover data. (Don’t commit yourself to unrealistic restore times, but if the new system can significantly improve restore times, show that!)

- A letter all ready to go, with which your boss is comfortable, that explains very matter-of-factly what

the company can expect if it doesn’t provide the funding you need.

Make a formal presentation

The more expensive your solution, the more important it is that you make a formal presentation, especially if you are in a corporate environment. A formal technical presentation has three parts: an executive summary, an overview section that goes into more detail, and a technical specifications section for those who are *really* interested:

Executive summary

: This should be one page and should explain on a very high level what is proposed in the rest of the presentation. Global figures and broad descriptions are good; do not go into too much detail. This is made for the VP, who needs to do the final sign-off but has 20 other presentations just like yours to review at the same time. Basically, state the current problem, and describe your solution.

Overview section

: Go into detail in this section. Use plenty of section headings, allowing your readers to read ahead or skim over it if they like. Headings also allow the people who read only the executive summary to look up any specific area they are not clear on. The outline of the general section should match that of the executive summary. You can include references to other publications, such as magazine reviews of a particular product, but do not quote them in detail. If they are relevant, you can attach copies in the technical section. Make sure you demonstrate that you have thought this through and that it is not just a stopgap measure. Put a high-level

comparison between the option you chose and the other options available, and explain why you chose the one you did. Describe how it allows for future growth and how much growth it will allow before you must reconsider. Also explain what plans you have for the old methodology and what conversion method you are going to use, such as running both in parallel for a while. Tables are also good. If you can use real numbers, it is much more effective: just make sure you can back them up. If anyone believes that the numbers are made up, it will totally invalidate the report. Try to compare the up-front cost of your solution with the surprise cost of lost data.

Technical specifications

Go wild. If anyone has made it this far, they're either really interested or a true computer techie just like you! If this report is the cost justification for a new backup drive, find a table that compares the relative cost per megabyte of all the various options. Include hard numbers and any white papers that are included with the proposed product. If you think it is relevant, but possibly too long and boring, this is the place to put it.

Good Luck

The chapters that follow explore in depth the various methods that you may employ to back up your systems, especially open-source tools. Most of these topics are also covered in documentation from the appropriate vendor or open-source team; this book is not meant to be a replacement for that documentation. Here, I try to explain things that are not covered in the documentation and possibly address some subjects more frankly than can a manual provided by the vendor.

Welcome to the world of backups.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Part 2. Open-Source Backup Utilities

Part II consists of five chapters:

Chapter 3, Basic Backup and Recovery Utilities

Describes all the basic backup utilities that you should know about, including `dump`, `cpio`, `tar`, and `dd` for Unix systems, `ntbackup` and `System Restore` for Windows systems, `ditto` for Mac OS systems, and the GNU versions of `tar`, `cpio`, and `rsync` that are available for all of these platforms.

Chapter 4, Amanda

Describes the popular open-source backup project of the same name.

Chapter 5, BackupPC

Discusses how to use this disk-based backup and recovery system.

Chapter 6, Bacula

Is a guide to the open-source backup product that is designed to scale from single machines to enterprise networks.

Chapter 7, Open-Source Near-CDP

Describes tools and techniques for achieving near-continuous data protection.

Chapter 3. Basic Backup and Recovery Utilities

Basic backup utilities are the backup utilities upon which all noncommercial backup systems are built. They accomplish the important task of copying data from one place to another, and usually copying into another format (for example, `tar`). None of these tools have any built-in scheduling abilities, nor can they make a catalog to keep track of the backups that you make with them. If you want to perform these tasks, you'll need some type of wrapper and scheduling application. This could be a simple batch script and a scheduled task on a Windows system, a shell script and `cron` entry on a Unix or Mac OS system, or one of the sophisticated open-source utilities covered later in this book.

Basic backup utilities include the native versions of `dump`, `cpio`, `tar`, and `dd` for Unix systems, `ntbackup` and System Restore for Windows systems, `ditto` for Mac OS systems, and the GNU versions of `tar`, `cpio`, and `rsync` that are available for all these platforms. Whether you're just starting out in the backup world or you're an experienced systems administrator, you need to be familiar with these utilities.

An Overview

This chapter describes the benefits and pitfalls of several utilities. For all versions of Windows since NT, `ntbackup` is the only native choice for a traditional backup application, although you should also be familiar with System Restore. Mac OS X users running a version greater than 10.4 have a number of Unix-based backup tools available to them, including `cpio`, `tar`, `rsync`, and `ditto`. For commercial

Unix systems, `dump` and `restore` are quite popular, but they're not considered a viable option on Linux. `dump` is available on Mac OS, but it doesn't support HFS+. After `dump` and `restore`, the native backup utility with the most features is `cpio`, but it is less user friendly than its cousin `tar`. `tar` is incredibly easy to use and is more portable than either `dump` or `cpio`. The GNU versions of `tar` and `cpio` have much more functionality than either of the native versions. If you have to back up raw devices or perform remote backups with `tar` or `cpio`, `dd` will be your new best friend. Finally, `rsync` can be used to copy data between filesystems on Windows, Mac OS, Linux, and Unix.

This chapter begins with an overview of each of these backup utilities. It then goes into detail about the syntax for each command for both backup and recovery. Finally, near the end of the chapter, you'll find an invaluable comparison chart that can be used as a quick-reference guide for comparing `tar`, `cpio`, and `dump`.

How Not to Use dump

I went on one gig to fix a client's "email" problem. Turns out it wasn't an email problem; it was a DNS problem. They also asked me to look at their backups. What I found was appalling. They were doing backups by issuing commands to run `dump` out of `cron`:

They didn't write a script; they just issued successive `dump` commands at different time intervals. Subsequent dumps were being executed before the previous one was finished.

They used the rewind device driver.

They were amazed they could fit everything on one tape!

How Mac OS Filesystems Are Different

Tip ✎

Leon Towns-von Stauber (the author of Chapter 14) contributed this information about Mac OS backups.

What can make Mac OS X backups tricky is the default native filesystem format, HFS+, which is the advanced version of the legacy Macintosh Hierarchical File System. There are significant differences between HFS+ and the Unix File System (UFS), including support for forks (multiple sets of data associated with a single file) and specialized file attributes (such as type, creator, and creation date). While Mac OS X can work with UFS filesystems, the UFS format is not nearly as commonly used as HFS+, nor as well supported by Apple and third-party software vendors.

A utility not designed to handle the unique features of HFS+ can cause backups to go haywire, losing essential forks and attributes, making full restoration impossible. The biggest problem is the resource fork, a set of auxiliary data associated with many kinds of Macintosh files. Despite being frowned upon by Apple since the release of Mac OS X, many applications still use resource forks to store information such as thumbnail icons for image files, and even Apple still uses them to store the contents of aliases, which are the GUI equivalents to symbolic links.

Before Tiger (Mac OS X 10.4), even the Unix-standard native utilities ignored forks and Macintosh attributes. If you're using Mac OS X 10.3 or earlier without third-party tools, your best options are CpMac (an HFS+-aware `cp` equivalent included with the Developer Tools), `ditto` (a recursive copying utility that supports resource forks and HFS+ attributes through use of the `–rsrc` flag), or `asr` (Apple System Restore, a volume cloning utility).

Due to the difficulty of making backups of Mac OS X systems before Tiger, a number of Mac OS X-specific variants of standard backup utilities sprang up on the Internet, including `hfstar`, `xtar`, `hfspax`, `rsync_hfs`, and `psync`, along with graphical frontends such as `RsyncX`, `PsyncX`, and `Carbon Copy Cloner`. Cross-platform applications such as Amanda and BackupPC also used these tools to support HFS+ backups.

cpio

`cpio` can be a very powerful backup tool. Its most important feature is its ability to accept the list of files to be backed up from standard input. It's the only native utility that can do this. This feature can be combined with the use of touch files and the `find` command to create incremental backups.

Unlike `dump`, however, `cpio` cannot:

- Perform incremental backups without the use of `touch` files and `find`
- Leave both `atime` and `ctime` unchanged after a backup (see the section "Don't Forget Unix mtime, atime, and ctime" in Chapter 2)
- Perform an interactive restore, like the `-i` option in `restore`

Why isn't cpio more popular?

If `cpio` is so powerful, why is `tar` more popular? One reason is that the basic operations of `tar` are much simpler (and more standard) than the same operations in `cpio`. For example, every version of `tar` supports `tar cf` *`device`* and `tar xf` *`device`*, whereas `cpio` sometimes supports the `-I`

and `-O` options and sometimes does not. If you add up all the `cpio` options available on all the various versions, you would find more than 40 of them. There are also some arguments that use the same letter but have completely different functions on different versions of Unix. Another reason why `tar` is more popular is the development of GNU `tar`. It combines the power of `cpio` with `tar`'s ease of use.

ditto

`ditto` is found only on Mac OS systems and is normally used to clone one disk to another; it is used in that fashion in Chapter 14. `ditto` can be also used to create a ZIP or `cpio` file. Because we use the tool in this book, and it's commonly used in Mac OS environments, it's covered in this chapter.

dd

The `dd` command is not a backup command used by most people. It is a very low-level command designed for copying bits of information from one place to another. It does not have any knowledge of the structure of the data it is copying—it doesn't need to. Therefore, unlike `dump`, `tar`, and `cpio`, it is not used to copy a group of files to a backup volume. It can copy a single file, a part of a file, a raw partition, or a part of a raw partition, and can even copy data from `stdin` to `stdout` while modifying it en route. Again, although it can copy a file, it has no knowledge of the filename or contents once it has done so. It simply copies the bytes that are in the place from which you told it to copy. It then puts those bytes where you told it to put them.

Although dd is rather simplistic, it is extremely flexible. It can copy files or partitions regardless of format. It can translate data between two different platforms, such as EBCDIC to ASCII, or big endian to little endian. (The concept of big endian/little endian is explained in detail in the section "The Little Endian That Couldn't" in Chapter 23.) A perfect example of dd's flexibility is the Oracle backup script included in Chapter 16. Oracle data is allowed to be in files in the filesystem or on raw disk partitions. Since the script could not predict which configuration each DBA would use, it used dd, because it could copy both files and raw partitions. That way the DBA can use whichever configuration makes most sense for his application, and the script will automatically back up either configuration. It even backs up a mixed configuration, in which some of the data sits on files and some sits on raw partitions. This is the kind of flexibility dd gives you.

dump and restore

dump and restore are considered by many to be the most powerful tools in the Unix backup toolbox. dump and restore's differentiating features include being able to back up files without changing their access time and being able to use a mini shell to interactively select the files you want to restore before you begin. dump and restore are relatively sophisticated commands, with simple interfaces whose essential options are the same on most Unix systems. There is a lot of controversy surrounding dump and whether or not it can properly back up an active filesystem. Read more about that in the dump section later in this chapter.

ntbackup

This is the only native tool in Windows that you can use to create a traditional backup, although some people do download and use GNU `tar` or `rsync` on their Windows systems. Like the Unix utilities covered in this chapter, it can back up to disk or tape, and you can specify a number of options. You can even save these options in a configuration file and then tell Windows to use that configuration file when `ntbackup` runs. The configuration file allows you to run automated backups with this tool.

rsync

Think of `rsync` as an open-source, fancier version of the Unix `rcp` command, that can be used to synchronize two folders even if they're on separate systems. Its basic syntax is essentially the same as `rcp`, so those familiar with that command should find `rsync` very easy to understand. Two of the open-source backup products covered in this book use `rsync` with other tools to provide backup and recovery functionality, so we'll cover its basic functionality in this chapter.

System Restore

System Restore isn't quite like the other tools in this chapter, but it's important to mention it. Since Windows 2000, you can use System Restore to create a snapshot of your system. It backs up a few critical files and your registry, allowing you to roll back your system state to a previous point in time.

tar

The greatest feature of `tar` is its wide acceptance, which is due in large part to its ease of use. Nearly everyone knows how to read a `tar` volume. If they don't, it's really easy to show them how. If it is a `tar` file on disk or even a compressed `tar` file, programs such as WinZip[1] can automatically decompress it and read what's inside. (WinZip cannot open a `cpio` archive.) It is also much more portable between Unix platforms than `dump` or `cpio`.[2]

If you need to make a quick backup of a directory or a set of files, it's hard to beat `tar`'s ease of use. However, if you need to make regular backups, you'll be looking for features that the *native* version of `tar` does not have. Among other things, you'll want to make incremental backups, leave `atime` alone, and make sure that you're restoring the proper permissions and ownership of files. To do these sorts of things, you can use GNU `tar`, or you can look at `cpio`.

> **Tip ✎**
>
> The explanations of the basic backup utilities that follow are not meant to replace the official documentation for those commands. You should definitely become familiar with the documentation for each command. It may contain anything from minor to major caveats for that particular OS. In some cases, vendors document an extra feature or two. Always stay up to date with the documentation for your backup command—whatever it is.

Other Utilities

This section contains a list of commands that we don't cover in this book for various reasons.

asr

`asr`, for Apple System Restore, is an imaging utility found only on Mac OS systems. It is used primarily as a bulk-cloning tool, similar to the way Windows customers use the `ghost` utility. It is an image-based utility and can be used to copy directly from one hard drive to another or to create a disk image of a hard drive, similar to an ISO file in other operating systems. Such a file carries a *.dmg* extension.

pax

The portable archive exchange, or `pax`, utility produces a portable archive that conforms to the Archive/Interchange File Format specified in IEEE Std. 1003.1-1988. `pax` also can read and write a number of other file formats such as `tar` or `cpio` and is used by the Mac OS install utility. Like many things in the Unix world, `pax` has a group of devoted followers that swear it's the best way to go. However, it will not be covered here because most people don't use it.

psync, rsyncx, hfstar, xtar, and hfspax

Since Mac OS X was built on top of a Mach Unix kernel, it shipped with a number of Unix-style tools such as `tar`, `cpio`, `pax`, `cp`, and `rsync`. Unfortunately, the early Mac OS versions of these tools did not support the concept of a multifork filesystem such as HFS+, and GNU `tar` didn't support it either.

`psync`, `rsyncx`, `hfstar`, `xtar`, and `hfspax` are all tools contributed by the Mac OS community that were designed to overcome the limitations of Mac OS's native tools. `psync` and `rsyncx` were written to behave like `rsync`, but to properly handle resource forks. `hfstar` and `xtar` behaved

like `tar` but handled resource forks. Finally, `hfspax` did the same thing for `pax`.

As of Mac OS 10.4.x, `tar`, `pax`, `cp`, and `rsync` all properly handle resource forks using the AppleDouble format. (According to Apple, these commands now use the same API as `Spotlight`, the Mac OS search tool.) When a file is copied into a format that doesn't support multiple forks, such as `tar`, `cpio`, or even a UFS filesystem on a Mac OS system, the tools mentioned here convert the file into two files. The first file contains the *data fork*, or actual data for the file. The second file is the *header file*; it stores the resource fork and finder information. The datafile is stored using the original filename for the file. The header file is the name of the file preceded by the string "._":

```
mydocument.txt
._mydocument.txt
```

When the multifork file is copied or restored from the nonmultifork format (`tar`, `cpio`, UFS) into a multifork format (HFS+), the two files are converted back to a single file with a data fork and a resource fork.

---

[1] WinZip is a registered trademark of Nico Mak Computing, Inc. You can download a demo version from http://www.winzip.com.

[2] The DJGPP project, a port of `gcc` and the GNU tools and utilities suites to MS-DOS and Windows, made `cpio` its portable archive standard and has ported both GNU `cpio` and GNU `tar` to DOS and Windows as 32-bit executables.

Backing Up and Restoring with ntbackup

The `ntbackup` command activates the `ntbackup` GUI and, unlike with all other commands covered in this chapter, you cannot select what to back up with the `ntbackup` command itself. You have to select that from the GUI; however, you can run the GUI once, select what files to back up, and save that to a *.bks* file you specify on the command line later.

> **Tip ✎**
>
> As with the other tools covered in this chapter, this section is not meant to replace the help page for `ntbackup`. It has many other options not covered here.

In addition to selecting which files are going to be backed up, you can also select values for a number of other options:

- Type of backup (normal, copy, differential, or daily)
- Type of target (disk or tape)
- Name of target (for example, *f:\backupfile.bkf*)
- Append or overwrite existing backups on target
- Logging level (verbose, summary, or none)

These options can be specified as options on the command line or in the `ntbackup` GUI and saved as part of a *.bks* file. However, since you have to run the `ntbackup` GUI to create an `ntbackup` setup, we won't cover the command-line switches in detail. Instead, we'll show you how to get Windows to automatically create the command you need to run.

Creating a Simple Backup Configuration

To create a simple backup with `ntbackup`, you need to create a backup options file using the `ntbackup` GUI, save it, then specify that options file when performing an `ntbackup` backup. Start the `ntbackup` GUI by typing `ntbackup` at the command prompt or by selecting Start→All Programs→Accessories→System Tools→Backup. From the Backup tab, select drives or directories to back up. Please note that you can back up the `System State` as well.

Next, you need to select various options about the backup. The two primary choices are the type of backup and where it will go. The available backup types are normal, copy, differential, and daily:

Normal (default)

: Back up the selected files and mark them as backed up.

Copy

: Back up the selected files but do not mark them as backed up.

Incremental

: Back up the selected files if they have changed since the last backup and do not mark them as backed up.

Differential

: Back up the selected files if they have changed since the last backup but do not mark them as backed up.

Daily

: Back up only the files that were modified today.

To select something other than the normal backup type, select Tools→Options→Backup Type. While you’re in the

Options dialogue box, browse the other tabs to see if you want to change any of those options as well. Click OK to close this dialogue box.

You then need to select whether or not you're going to use disk or tape. Disk is probably the best option for a simple backup, especially if you just want to back up to a share that's going to be backed up by another process. You then need to select a filename for the backup file. Once you've selected these options, select Job→Save Selections As, and save the options to a filename that you record, such as *c:\mybackup.bks*.

Executing Your Simple Backup

To run the backup you created, you've got three choices. The first choice is to simply click Start Backup in the `ntbackup` GUI. You can also run it from the command line if you've saved the options to a file. The following command assumes that you didn't select any options other than which files to back up and specifies all of the important options as arguments to the `ntbackup` command. It backs up the files you selected and saved as *c:\mybackup.bks*, gives the job the name "Daily Backup," and backs the data up to the file *F:\backup.bkf*.

```
C:\ ntbackup backup "@C:\mybackup.bks" /M Normal /J "Daily Backup" /F
"F:\backup.bkf"
```

The next choice is to create a scheduled task with this command in it. If you'd rather let Windows figure out all the command-line switches for you, you can simply use the `ntbackup` wizard to create the scheduled task. Once you've opened `ntbackup`, select the Schedule Jobs tab, select a date on the calendar, and click Add Job. Select the items you want to back up in the "Items to Back Up" dialogue box. The next dialogue box asks you to select a destination

directory and filename, and the next screen asks you to select a backup type. The following screen gives you some other options, including whether or not to verify the data after it's been backed up. You can then specify whether or not this backup should append to or overwrite any backups already on the destination. Finally, you're asked to name the job and create a schedule of when it should run. Once you've done that, Windows creates a scheduled task with the appropriate commands in it. The one I created during my example looks like this:

```
C:\WINDOWS\system32\ntbackup.exe backup "@C:\mybackup.bks" /a /d
"Set created 3/12/2006 at 8:35 PM" /v:no /r:no /rs:no /hc:off
/m normal /j "mybackup" /l:s /f "C:\Backup.bkf"
```

Tip ✎

`ntbackup` can also be used to back up and recover Exchange. See Chapter 20 for more details.

Restoring with ntbackup

You cannot restore from the command line using `ntbackup`. What you can do is start `ntbackup` and select the "Restore and Manage Media" tab. Displayed in this window is a list of backups that `ntbackup` knows about. You can select any of the backups in this dialogue box, and you'll be presented with a tree of the files that are in that backup. You can then select which files you want to restore, decide whether or not to restore the files to their original location or another location of your choosing, and tell `ntbackup` to restore them by clicking Start Restore. You're then given a choice to select advanced options; the restore starts when you click OK. It really doesn't get much easier than this!

Using System Restore in Windows

Anyone who has used Windows for a significant amount of time has had the experience of installing a new piece of software and having it render their Windows system useless. Previously, the only option would be to reinstall Windows and all your applications, but with System Restore this is no longer the case. If you're able to boot into safe mode and select System Restore, you'll probably be able to find a stable version of Windows to restore to. You'll be back up and running in no time!

> **Tip ✎**
>
> System Restore is a bit different from the other utilities in this chapter because it doesn't create a backup in the traditional form, and you can't use it as part of another tool. However, it's a very important recovery tool that ships with Windows XP and later, and you should become familiar with it.

System Restore in Windows XP and later backs up the Windows registry and critical files to create a *restore point*. Windows automatically does this when it deems you are about to perform a significant event, such as the installation of a new driver or major patch. In addition, you can create your own restore points whenever you want, or at automated intervals using a scheduled task. You can then use any of the restore points that you or the system created to restore your system state to a previous point in time.

Creating Restore Points

As mentioned previously, Windows actually creates a lot of restore points for you, assuming you haven't disabled System Restore. To check whether System Restore is enabled, log in as a user in the Administrators group, and select Start→My Computer→Properties, and select the System Restore tab. You can then enable or disable it from this tab.

> **Tip ✎**
>
> You must be logged in as Administrator or be in the Administrators group to use System Restore.

Anyone in the Administrators group can create a restore point at any time by selecting Start→All Programs→Accessories→System Tools→System Restore→"Create a restore point." A dialogue box asks you to name the restore point you're about to create. You can call it anything, such as *Just before I Install Doom*. The system then creates the restore point and gives it that name. You can then restore Windows to that point in time using System Restore.

> **Tip ✎**
>
> You could also run System Restore by running the command `%SystemRoot%\system32\restore\rstrui.exe`, but it's not likely you'll remember that one.

If you don't want to trust Windows to create restore points for you, and you don't want to manually create one when you need one, you can create a scheduled task to create one for you as often as you would like. Select Start→All Programs→Accessories→System Tools→Scheduled

Tasks→Add Scheduled Task. Click Next, and select System Restore in the next dialogue box. Select how often you want to run it and when you want it to run, and enter a username and password of a user in the Administrators group. Windows then creates a restore point with your specifications.

Recovering Windows Using a Restore Point

If your version of Windows has become unstable due to a recent patch or driver installation, you need only select System Restore, select a previous point in time, and tell it to restore Windows to that point in time. If Windows is truly unstable, the hardest part may be getting Windows to boot at all. The best way to do this may be to boot into safe mode and log in as Administrator.

Once you have Windows running in any way, select Start→All Programs→Accessories→System Tools→System Restore, and select “Restore my computer to an earlier time.” You’ll then be presented with a dialogue box like the one shown in Figure 3-1.

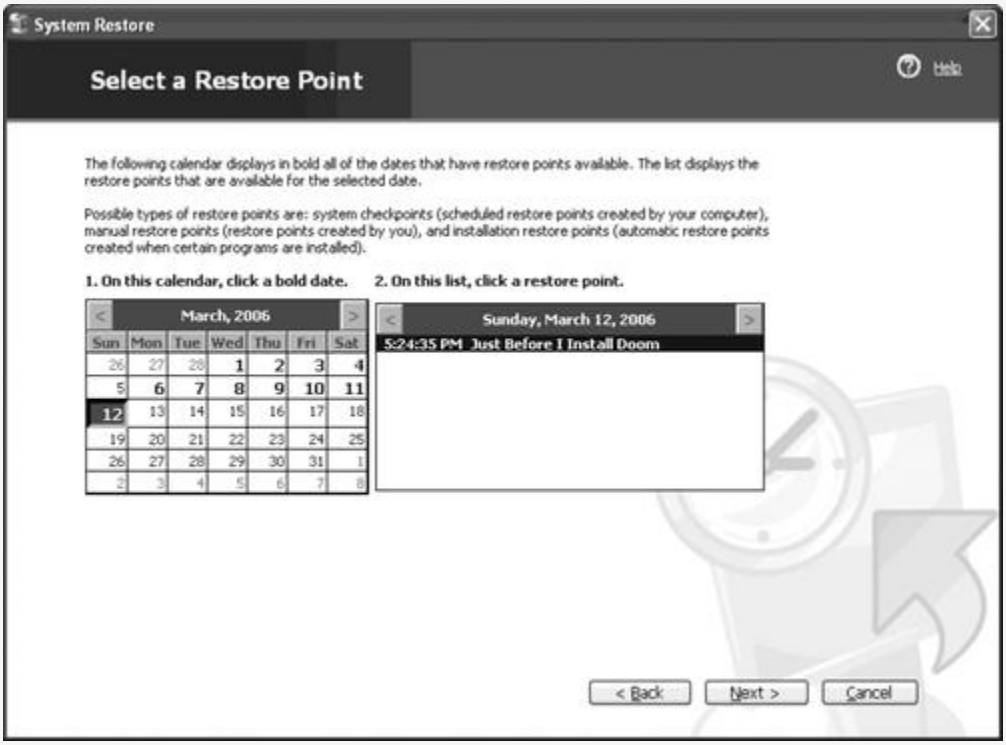


Figure 3-1. Selecting a restore point

The most recent date with a restore point is automatically selected on the calendar, and the restore points from that date are displayed to the right. You can restore to that point, or you can select an earlier date if you believe the most recent date to be suspect as well. Now select the restore point you want to restore to, and click Next. Windows asks you to confirm your choice, of course, and warns you to save any data and close any open programs because this restore requires a reboot.

The rest is a matter of clicking Next until it’s done, rebooting, then testing the restored version of Windows to see if your problems have been fixed. If so, you’re done. If not, just go through the process again until you find a restore point that works for you.

Backing Up with the dump Utility

For many environments, `dump` may be all you need to ensure good-quality backups. There's a lot of controversy surrounding `dump`, though, stemming from the fact that it doesn't access the data through the filesystem the way most other backup utilities do. `dump` accesses the filesystem device directly. This is why it can back up files without changing their access times. However, it's also why the manpages for `dump` have always said to unmount filesystems prior to backing them up. Of course, no one ever does that, hence the controversy.

dump on Mac OS and Linux

Linux administrators should be aware that `dump` is not considered a good way to back up a Linux system, and `dump` doesn't support the HFS+ filesystem in Mac OS. RedHat officially deprecated `dump` in RedHat 9, and the following quote from Linux Torvalds sums up the Linux community's attitude towards `dump`:

"`dump` simply won't work reliably at all even in 2.4.x: the buffer cache and the page cache (where all the actual data is) are not coherent. This is only going to get even worse in 2.5.x, when the directories are moved into the page cache as well. So anybody who depends on `dump` getting backups right [on a Linux system] is already playing Russian roulette with their backups. It's not at all guaranteed to get the right results—you may end up having stale data in the buffer cache that ends up being "backed up".... `dump` may work fine for you a thousand times. But it will fail under the right circumstances. And there is nothing you can do about it."

You will have to make up your own mind on whether or not `dump` is right for you, but apparently `dump` isn't the best way to back up a Linux system.

`dump` and `restore` are available on Mac OS, but they work only with the UFS filesystem. There is no `hfsdump` for the HFS+ filesystem, and I know of no plans to create such a tool.

To use `dump` and `restore` for regular system backups, you need to understand the following:

- How to use `dump` to back up a filesystem (with the appropriate options)
- How the backup ends up on the volume
- How to get the table of contents of a `dump` volume
- How to manipulate the volume and restore from a backup created by `dump`
- The limitations of `dump` and `restore`
- What you should be doing if you are using `dump` on a regular basis

The first thing to understand is what your `dump` command is and what its options are. See Table 3-1 for a listing of `dump` commands on various Unix versions. The following section is essentially a unified manpage for these `dump`-like commands on specific operating systems.

Warning ☠

Although there is a `dump` command on Mac OS, it does not support the HFS+ filesystem, which is the most common filesystem type on Mac OS.

Table 3-1. dump-like commands on different Unix versions

Unix version	Command
HP-UX 9.x/HP-UX 10/SunOS/IRIX	`(r)dump`
Solaris	`ufsdump`
SCO	`xdump`
Network Appliance	`dump`
AIX	`backup` and `rdump`
Linux	`dump`
SGI	`dump` and `xfsdump`
Tru64 Unix	`dump` and `vdump`
Linux/Mac OS	See the sidebar “dump on Mac OS and Linux”

Syntax of the dump Command

Let’s start with the basic `dump` command:

```
#  dump levelunbdsf blkg-factor density size device-name file_system
```

The following are examples of running this command:

- To create a full backup of */home* to a local tape drive called */dev/rmt/0cbn*:

```
#  dump 0unbdsf 126 141000 11500 /dev/rmt/0cbn /home
```

- To create a full backup of */home* to an optical or CD device called */backup/home.dump*:

```
#  dump 0unbdsf 126 141000 11500 /backup/home.dump /home
```

- To create a full backup of */home* to the remote tape drive */dev/rmt/0cbn* on *elvis*:

```
#  (r)dump 0unbdsf 126 141000 11500 elvis:/dev/rmt/0cbn /home
```

The preceding commands use three options (`0`, `u`, and `n`) that do not require arguments and four options (`b`, `d`, `s`, and `f`) that require a "companion" argument.

The `dump` command accepts as its first argument a list of options, then each option's argument is placed on the command line in the same order in which the options are listed. Figure 3-2 illustrates how the `dump` command options relate to their companion arguments.



Figure 3-2. Sample dump command

The Options to the dump Command

The `dump` utility has seven main options that are available on most platforms:

`0-9`

Specifies the level of backup that `dump` should perform.

`b`

Specifies the blocking factor that `dump` should use.

`u`

Tells `dump` to update the *dumpdates* file.

`n`

Tells `dump` to notify the members of the Operator group when a `dump` is completed.

`d` and `s`

Tells `dump` how large the backup volume is. `dump` uses these numbers to estimate how much "tape" is available.

`f`

Tells `dump` what device to use.

`W`, `w`

Tells `dump` to perform a dry run that tells you what filesystems need to be backed up (these are seldom used).

If you are using `dump` for regular system backups, you should be using most of the preceding options. It is important to note that many of these options have default values, eliminating the need to specify that option and its argument in the `dump` command. For example, the default backup level is usually 9. The problem with the default values is that they vary between operating systems and may also vary even on the same operating system, depending on factors such as media type. It is better to specify each of these options the same way on all your `dump` backups to simplify making restores at a later date.

dump and restore Save the Day

It had been a long, hard week, and we were trying to finish up a few things so we could go home. That's when we got the call. That's *always* when you get the call. A very important directory, which contained a seldom used but essential utility, was missing from the system. "No problem," I said, "we've got it on tape." Or so I thought. When I went to recover the files, I realized that this directory had been missing for a while. In fact, it had been missing for so long that it had not been backed up by the commercial utility we were using. You can imagine the feeling that was in my stomach.

I looked over at the old filing cabinet where we kept a pile of poorly organized, inadequately labeled, and almost forgotten `ufsdump` tapes. At that moment, they were the

most important tapes in the world, because they had been made before we started using the commercial utility. I put those tapes in the drive, one by one, using the `table of contents` option of `ufsrestore`, in hopes that one of them would be the right one. The stack was getting shorter and shorter. Finally, one of the tapes looked like it could be the one. I switched modes, using the `interactive` option, and there it was. I selected the directory and extracted it. The directory was saved, and the customer never even knew that we almost weren't able to restore the data. That was one day I was really glad that I knew `dump` and `restore`. (I also learned how important it is to archive monthly full backups.)

Specifying a complete or incremental backup (0–9)

The first argument that you can specify is the dump level; you can use any number from 0 to 9. (See Chapter 2 for an explanation of backup levels.) Incremental dumps refer to the *dumpdates* file for the date of the last lower-level backup. (This file is discussed in the section "Updating the dumpdates file (u)" later in this chapter.) For example, if you are performing a level 5 backup, `dump` backs up all files that have changed since the last backup that was level 4 or lower. It gets the date of this backup from *dumpdates* (usually */etc/dumpdates*). Since the *dumpdates* file is needed for incremental backups, you must use the `u` option to update it.

Specifying a blocking factor (b)

The `b` option specifies the number of blocks to write in a single output operation. This refers to the *number* of physical blocks. The *size* of the entire block that `dump` writes depends on the size of the physical block multiplied by the blocking

factor. For most versions of Unix, the physical block size for `dump` is 1024 bytes. So, if you specify a blocking factor of 10, the size of the actual block that `dump` writes is 10,240, or 10 K. This option is not available on SCO.

> **Warning ☠**
>
> At least one flavor of Unix allows you to change the blocking factor for `dump` but not for `restore`. This means that you can create `dump` volumes that you can't read! Make sure that your flavor of `restore` allows you to change the blocking factor.

Updating the dumpdates file (u)

The u option causes `dump` to update the *dumpdates* file for the filesystem that you backed up. (The *dumpdates* file is usually */etc/dumpdates*, but is */var/adm/dumpdates* on HP-UX 10.x.) This is a plain-text file that lists each filesystem's raw device and the date that the last backup of each level was taken on that device. Here is an example */etc/dumpdates* file taken from a Solaris box:

```
/dev/rdsk/c0t1d0s0              0 Sun Apr 30 23:07:22 2006
/dev/rdsk/c0t1d0s0              1 Wed May  3 02:49:51 2006
/dev/rdsk/c0t3d0s0              0 Sat May 20 00:31:49 2006
/dev/rdsk/c0t3d0s0              1 Mon May 29 01:33:33 2006
/dev/rdsk/c0t3d0s0              5 Wed May 31 00:28:14 2006
```

You can see that device *c0t1d0s0* had a level 0 backup on April 30, and a level 1 backup on May 3, 2006. Device *c0t3d0s0* had a level 0 backup on May 20, a level 1 on May 29, and a level 5 on May 31.

There are a few important things to note about the *dumpdates* file. The first time you run `dump` on a system, you must first create an empty *dumpdates* file, and it must be owned by

root. If it is not there or is not owned by root, `dump` does not create it. Your dump continues, but it will complain. Note that *dumpdates* is updated only if the entire dump completes successfully. If any errors cause `dump` to abort, *dumpdates* is not updated. This means that it is a good file to use for an automated script that checks to see if your dumps worked. The following list shows the various names and locations of the *dumpdates* file:

- HP-UX 9.x, SunOS, Solaris, AIX, Linux, IRIX: */etc/dumpdates*
- HP-UX 10.0: */var/adm/dumpdates*
- SCO: */etc/ddate*

You might not want to use the u option when making a special “one-time” backup volume, because doing so changes the behavior of other backups. For example, if you are making a one-time level 0 backup for someone and use the u option, your automated level 1 backups will reference that level 0 backup that has been given to someone else and is not a part of your normal backup pool.

Tip ✎

The *dumpdates* file, whatever it may be called, can be viewed or modified with a standard text editor. You might want to do this, for example, if you know that this week's level 0 backup has been eaten by a hungry tape drive. You don't have time to rerun a full level again, but you want some sort of backup. However, if you run a level 1, it references this week's level 0 backup, which you know is no good. You can edit the level 0 line for the appropriate filesystems, changing the date to the date of last week's level that has not been eaten. Your level 1 then references last week's level 0 rather than this week's level 0, which was destroyed. This can allow you to sleep a little better after that level is destroyed, without having to rerun a complete level 0.

Notifying your backup operators (n)

The `n` option causes `dump` to notify everyone in the operator group, as specified in the */etc/group* file, if a `dump` backup requires attention. This notification looks similar to a `wall` message. (This option is not available on SCO.) A `dump` backup may require attention when any of the following occurs:

- A `dump` backup reaches the end of a tape, or your CD fills up.
- A backup drive malfunctions, causing write errors.
- There are difficulties reading from the disk drive.

Specifying density and size (d and s)

The density (d) and size (s) options do not affect *how* data is written to the backup media. The dump command uses them only to determine how much data can fit on a given volume and to determine when it has reached the logical-end-of-tape (LEOT, or the point at which dump thinks the volume is full) before it reaches the physical-end-of-tape (PEOT). dump then prompts the operator to switch volumes. The logic behind this is to keep the volume from hitting PEOT, because older versions of dump do not handle this well. Here is a quick explanation of these two flags:

d (density)

By specifying a density, you are telling dump how much data fits on one inch of tape. (This value is really a throwback to the nine-track tape days, but dump uses it in combination with the s option to figure out how large the backup volume is.) If you want to make sure that dump uses the entire volume, use a large value such as 80,000.

s ("tape" size in feet)

This option tells dump how long the tape is. It then calculates how much data fits on the tape using the values provided for size and density. If you want to make sure that dump uses the entire volume, use a large value like 500,000. Using 80,000 as the density and 500,000 as the size effectively tells dump that your volume is capable of storing 480 GB! (Yes, this and the d option both seem silly if you're backing up to disk or CD, but they are important. See the following section "Do I have to use the s and d options?" for more information.)

In actual practice, these options are very difficult to use and yield very little value. Most people fake out dump using values that make dump think it will never run out of tape. This causes dump to use the entire volume and lets it

discover the PEOT if or when it gets that far. There are many reasons for this:

- The `dump` command can now detect and handle PEOT (`dump` used to abort upon reaching PEOT). In Solaris, they even have an option that causes the tape to eject, and if you are using an autochanger, it then inserts the next tape. On Solaris, therefore, `dump` could then continue without intervention.
- The calculations work only if it's the only backup that `dump` has put on the volume. (For example, each time you use `dump`, you tell it the tape is 10,000 feet long. If you have already put at least one backup on the volume, it's no longer 10,000 feet long).
- If you were to use "real" values, you would probably have a small density value with a very large size value. Many Unix versions tell you that doing this can cause problems. (I'm serious. You have to make them up!)
- If you want `dump` to actually stop before PEOT, you need to underestimate the values, which results in using less space than the volume actually has. (Some budgets necessitate using every inch of every volume that you paid for.)

Adding compression into the calculation really complicates the process, since compression is one area in which the phrase "your mileage may vary" really applies.

Avoid Creating a dump Backup Across Multiple Volumes

By "across multiple volumes," I mean that this is a single `dump` backup that starts on one volume, runs until it hits LEOT or PEOT, and then continues on another volume. For example, if you have a 4 GB DDS tape drive and are backing up a 2 GB filesystem and a 3 GB filesystem, the

first dump backup would fit on the tape. The second one would fill up the rest of the tape, requiring you to insert a second tape to allow dump to finish (see Figure 3-3).

In my opinion, creating a backup in this manner is asking for trouble. If you have no choice, then you must do it, but it raises some questions and adds difficulty to your restore. For example, you have to load tape 1 and start reading it before you can load tape 2. It's already hard enough to do a restore in the first place! Also, I start wondering about how safe the files are that are stored near the end of the first tape. Are you sure they're safe? The dump command can be funny sometimes.



Figure 3-3. Example of a multiple-volume dump backup

Do I have to use the s and d options?

A few newer versions of dump have done away with these options and provided a new `size in kilobytes` option you can use to specify the size of the volume in kilobytes. Even so, I personally use the `s` and `d` options with every dump command I run so that I don't have to remember how different versions work. You will find this is a common theme throughout this book: the more things you can do the same everywhere, the fewer things you have to worry about. The more per-host and per-OS customization you do, the more trouble you can get into. (For example, the `size in kilobytes` option uses a different letter on each version of Unix that supports it!) In this case, using the archaic size and density options actually makes writing shell scripts much easier, because you can use the same options on most versions of Unix.

What happens, then, if you don't use the `s`, `d`, or `size in kilobytes` options? On some Unix flavors, `dump` uses the default values for size and density (except for AIX, which has apparently done away with these options altogether). Unfortunately, the default values are usually set to work with a nine-track tape. (Solaris has changed its default values to be slightly more sensible.) If this happens, `dump` will think it needs several volumes. The output of `dump` looks something like the following:

```
DUMP: Estimated 5860 blocks (3006KB) on 39.00 tapes.
```

Notice that it thinks it's going to need 39 tapes. This is what can happen if you do not use the size and density options to specify the capacity of the volume. As mentioned before, you can easily disable this feature by setting these values to some ridiculously high figure so that `dump` never thinks that it has run out of tape. (I personally use numbers like 1,000,000 for both.)

Specifying a backup device file (f)

The `f` option specifies the name of the backup device to which you are sending the data. (This "device," of course, could be either an actual tape device or a file sitting on a disk, optical platter, or CD.) If you are expecting to use the hardware compression feature of your tape drive, make sure that you choose a device that supports compression. If you want to send the data to a drive on another system, use the format *`remote_system_name:device`*. Most versions of Unix support using remote devices in `dump`, as long as you're alright with using `rsh` as an authentication mechanism.

Warning ☠

The use of `rsh` and */.rhosts* files is a major security hole, and many sites no longer allow their use! Don't go creating */.rhosts* files everywhere and blame it on me. Make sure you investigate whether you are allowed to use `rsh` at your site before you start using it. If you are not allowed to use `rsh`, you might want to look at implementing `ssh` as a drop-in replacement for `rsh`. See the section "Using ssh or rsh as a Conduit Between Systems" near the end of this chapter for more information.

Remote devices require that the host with the remote device trust this host via the */.rhosts* file. If you try to use a remote device from a nontrusted system, you might get the dreaded message:

```
Permission Denied
```

To test if you are a trusted host, try issuing the following command as root:

```
# rsh remote_system uname -a
```

If it does not work, you need to put a line with this system's name in the remote system's *~root/.rhosts* file.

Unfortunately, in today's mixed environments, you don't always know what other systems think a particular system's name is. The remote system might be using DNS, NIS, or a local *hosts* file. When you `rsh` to a system, it initially sees you as an IP address. It then does a `gethostbyaddr()` and tries to resolve that address into a name. Depending on how your particular system is set up, it may consult DNS, NIS, or the local */etc/hosts* file; the order in which it consults these sources also varies with your

setup. If it uses the local *hosts* file or NIS for address resolution, it may or may not appear with a fully qualified domain name such as *apollo.domain.com*. If it uses DNS, it appears with the fully qualified domain name. It is important to know this because this is the name you must put into the */.rhosts* file. Suppose your system is called *apollo*, and the remote system is *elvis*. If you want to `rsh` from *apollo* to *elvis*, you should try the easy step first. On *elvis*, enter this command:

```
$ echo apollo >>/.rhosts
```

If that doesn't work, *apollo* appears as something else to *elvis* (e.g., *apollo.domain.com*). To find out for sure, you can telnet to *elvis* from *apollo*, then use commands such as `last`, `who`, `tty`, or `netstat` to look at the field that lists the system from which you came. If it turns out to be *apollo.domain.com*, put that into the */.rhosts* file on *elvis*. (For example, at one client site, it appears as *apollo.DOMAIN.COM*.) Once you have put the correct name in */.rhosts*, `rsh` should work.

Displaying which filesystems need to be backed up (W and w)

The `W` and `w` `dump` options are available on most Unix systems and display information about which filesystems need to be backed up. Usually, the `w` option displays information on all filesystems, while the `W` option lists only those filesystems that need to be backed up, based on the backup level you have chosen. These options have slight variations between Unix flavors, so read the appropriate manpage.

Interesting options for Solaris's ufsdump utility

Solaris's `ufsdump` has a few options not found in other versions of Unix. It supports the `l` (autoloader), `o` (offline),

a (archive file), and v (verify) options:

l

The autoloader option ejects the tape if it reaches PEOT before dump is done. It then waits up to two minutes for the next tape to be inserted. This works well with sequential autoloaders.

o

The offline option merely ejects the tape at the end of the backup, protecting the tape from being overwritten by another process.

a

The archive file option writes dump's table of contents to *archive_ file* (as well as writing it to the volume, as all dump commands do). This file can then be used by ufsrestore to see if a file is on a given volume without having to mount that media.

v

The verify option compares the backup to the actual filesystem. While this may sound good in theory, it requires the filesystem to be unmounted, which is not practical in many applications.

What a dump Backup Looks Like

This section explains one primary difference between dump and its cousins, tar and cpio. dump writes a table of contents at the beginning of each volume while tar and cpio do not.

dump records an index on the volume

The index is read during an interactive restore, allowing you to run commands such as `cd` and `ls` on this table of contents, viewing and selecting files that you want for the restore. (The `restore` utility is discussed later in this chapter.) This interactive restore feature is one of `restore`'s biggest advantages over `tar` and `cpio`. Note one important thing about this index: it is made at the beginning of the backup, before it has tried to actually back up anything. The presence of the index makes the interactive restore efficient because you don't have to read the whole volume before you can see what's on it. However, the fact that it's created before the backup data is written, and possibly minutes or hours before the data is written to tape, means that files made during the backup are not included, and files deleted during the backup are listed on the index but are not actually on the volume.

Using the index to create a table of contents

You can create a table of contents of a `dump` volume by physically reading the contents of the index that `dump` creates and seeing what `dump` intended to write to the volume. Also, it is important to mention that this reading of the volume in no way guarantees the integrity of the actual file on the volume any more than an `ls -l` on a file in a directory verifies its integrity. You may be wondering why this discussion is included here, in the section about `dump`; it is because making this table of contents should be a part of every `dump` backup that you take. Having said that, how do you create a table of contents of a dump file? First, what does "dump file" really mean? Perhaps an illustration would help; see Figure 3-4.

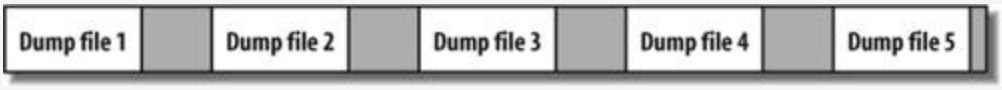


Figure 3-4. The format of a dump tape

A volume created by `dump` may have multiple dump files, sometimes called *partitions*, on it. Each file ends in an end-of-file (EOF) mark, symbolized in Figure 3-4 by shaded areas.

You have two options if you want to obtain a table of contents for dump file 3 in Figure 3-4:

- You can tell `restore` to read the third file on the tape using the `s` option; this causes `restore` to skip files 1 and 2 and read file 3. (This option does not apply to disk-based `dump` backups.)
- You can manually position the tape (using `mt` or `tpctl`) so that it is sitting at the beginning of that file, then tell `restore` to read it as if it were the first file on the tape.

> **Tip ✎**
>
> You must know the blocking factor in which the volume was written. If you are not sure, try the default by not specifying a blocking factor. If that doesn't work, see the section "How Do I Read This Volume?" in Chapter 23.

The first method is the easiest, because it involves only one step. The syntax of the command is as follows:

```
$ restore tsbfy file blocking-factor device
```

To read the third dump file on the tape with a blocking factor of 32, use the following command:

```
$ restore tsbfy 3 32 /dev/rmt/0cbn
```

Here's a list of the options used and what they do:

- The `t` option tells `restore` to read the volume index and provide a table of contents.

- The `s` option, and its accompanying argument `3`, tells `restore` to read the third dump file on a tape.
- The `b` option, and its accompanying argument `32`, tells `restore` that you used a blocking factor of 32 when you wrote this dump file.
- The `f` option, and its accompanying argument `dev`, specifies that the dump file is on that device.
- The `y` option tells `restore` to continue in the case of errors, instead of asking you if you want to continue.

If you do choose to manually manipulate the tape, as in the second option, you need to be familiar with your Unix version's magnetic tape command. This is usually `mt`. It has five options—`status`, `rewind`, `offline`, `fsf`, and `fsr`—four of which you might use when manipulating `dump` tapes. The format of the command is:

```
$ mt -t device argument
```

Tip ✎

If you are planning to position the tape, make sure you are using a *nonrewinding* device, such as */dev/rmt/0n*. Otherwise, it rewinds as soon as you finish positioning it!

Some versions of `mt` use a `-f` instead of a `-t`. The *`device`* argument is the no-rewind tape device that you are using, such as */dev/rmt/0n*. Now specify one of the following for *`argument`*:

`status`

This gives you the `ioctl` status of the tape device. It does not require an accompanying argument.

`rewind`

This rewinds the tape to the beginning. This option is spelled `rew` on some versions of Unix. It does not require an accompanying argument.

`offline`

This ejects the tape from the tape drive. This option is spelled `offl` on some versions of Unix. It does not require an accompanying argument.

`fsf` *x*

This is short for “forward space file.” It positions the tape forward *x* file marks, where *x* is a number greater than 0. (If you do not specify a value for *x*, it defaults to 1.) If you are at the beginning of the tape, you are at file 1, so if you want to be at file 3, you need to go forward two files. This requires an `fsf 2`, as in `mt -t` device `fsf 2`.

`fsr` *x*

This is short for “forward space record,” and is not needed when manipulating `dump` tapes. (If you do not specify a value for *x*, it defaults to 1.)

The following are examples of how to use the `mt` command. To rewind the tape */dev/rmt/0cbn*, issue the command:

```
# mt -t /dev/rmt/0cbn rewind
```

To fast-forward the tape */dev/rmt/0cbn* to the second file on the tape, issue the command:

```
# mt -t /dev/rmt/0cbn fsf 1
```

To eject the tape */dev/rmt/0cbn*, issue the command:

```
# mt -t /dev/rmt/0cbn offline
```

To get the status of the tape */dev/rmt/0cbn*, issue the command:

```
# mt -t /dev/rmt/0cbn status
```

Once you have positioned the tape to the proper file, simply use the same `restore` command as before, leaving off the `s` option and its argument:

```
$ restore tbfy 32 /dev/rmt/0cbn
```

Whichever method you use, the table of contents is sent to standard output, which you should redirect into a file. One important thing to note about this output is that the name of the filesystem dumped to this volume is not in the output. This table of contents is relative to that filesystem, whatever its name was. For example, if you backed up */var*, and you were looking for */var/adm/messages*, the output would look something like this:

```
345353  ./adm/messages
```

I recommend that you create a table of contents for each `dump` volume when you make it and store this output in a file that matches the name of the volume. Obviously, you should use a unique name, like:

```
./dump.system.filesystem.level0.May19.2006
```

Saving tables of contents in this way is very handy when you're searching for a file and you can't seem to find it on any volume. A quick `grep` of all the dump files shows you which volume you need.

A Day Late and a Dollar Short

I was once told that data needed to be recovered from a machine that had been decommissioned 10 years earlier. I was told the name of the machine and about where the tapes were stored, so I started digging.

When I found the tapes, once I scrounged a tape drive with low enough density to be able to read them, I discovered that they were in a `dump` format that was no longer supported! I found the source code for the original `restore` program (the BSD 4.1 one in this case), downloaded it to my machine (SunOS 4.0.1 in this case, a BSD 4.3-like system), and started working on porting the old program. No good. I soon realized it would take me weeks to do it; the filesystem and `dump` formats had changed that much.

There *had* to be a different way, so I searched the data vaults for more tapes. Luckily, I found another stack of tapes, marked as being in `tar` format. I had lucked out! Most of these tapes were still readable, and the data came off the first try.

Moral of the story: when you decommission a machine, make an archival copy of the data in every format you can, on every type of media you can. Some, like `dump`, are very efficient but might not be supported someday, while others, like `tar` and `cpio`, have stayed around year in and year out. Times change, media changes, formats change, so make as many variations as you can so your data will be retrievable for as long as possible.

This made me a big fan of using `tar` for archival purposes, but that makes excellent sense. Its name stands for Tape ARchiver, after all.

Doug Freyburger

Restoring with the restore Utility

While writing this section, one phrase kept coming to mind, from a commercial for a motion-sickness medication in the U.S. called Dramamine. "The *time* to take Dramamine is too *late* to take Dramamine." (By then, you're already sick.) The same thing applies to learning how to use the `restore` utility. You need to become very familiar with the various ways in which you can use `restore` to retrieve data from a backup created with `dump`. If you are in the midst of a critical restore as you read this, don't worry: this section is organized with that scenario in mind and includes every trick available in `restore`.

> **Tip ✎**
>
> This next section assumes that you know the volume was made with `dump` and that you know its block size. If you do not have this information, see the section "How Do I Read This Volume?" in Chapter 23.

Is the Backup Volume Readable?

To make sure that you know the format and block size of a tape, try listing its table of contents. The following command produces the table of contents of a volume created with `dump`:

```
$ restore tbfy block_size device-name
```

For example, to read the table of contents of a `dump` tape (made with a blocking factor of 32) on */dev/rmt/0cbn*, issue the following command:

```
$ restore tbfy 32 /dev/rmt/0cbn
```

If that works, then the rest is easy. (If not, read “How Do I Read This Volume?” in Chapter 23.)

Blocking Factor

Sometimes `dump` can write in a blocking factor that `restore` cannot read. This problem is usually very simple to get around. Once again, you need the block size in which the volume was written. Determine the volume’s block size as discussed in Chapter 23. Let’s assume that the block size of the volume is 65536. Use `dd` to read the volume, and pipe the output of `dd` to `dump`, giving “-” as the file argument. This tells `restore` to read its data from standard input.

```
# dd if=device-name bs=64k|restore tfy -
```

Why does this work? The blocking of data while writing to a volume drive actually changes how the data physically resides on the volume. The `restore` command needs to understand the blocking format to be able to read the volume. However, if you use `dd` to read the data from the volume, the data is put into a pipe. The `dd` command effectively sets the block size of the pipe to 1, allowing `restore` to use any block size when reading it.

Byte-Order Differences

The `dump` backup format is very filesystem-specific. If you have byte-order differences, the versions of `dump` and `restore` are probably also different. The easiest, and possibly the only, thing to do is to find a system that has the same operating system as the one that made the volume. That is because reversing the byte order may allow you to read the dump header but, depending on the `dump` format, it may render the restored files useless.

Different Versions of dump

Unfortunately, this issue only gets worse with time. Unlike the other utilities covered in this chapter, the `dump` command is tied heavily to the filesystem, and `dump` generally works with only one type of filesystem. The problem with this is that Unix vendors keep trying to improve the filesystem, so many Unix vendors have more than one type of filesystem. If `dump` exists at all on your version of Unix, it may support only the older filesystem types. In some cases, there are multiple versions of `dump`. For example, IRIX has both `dump` and `xfsdump`. Each version of `dump` also has its own version of `restore`. Different versions of `restore` may or may not be able to read a backup written by another version of `dump`. This is yet another area where your mileage will definitely vary.

Probably the best example of the changing nature of `dump` is SGI's XFS filesystem and its `xfsdump` command. On the surface, it looks like the old (`efs`)`dump` command with a few new options. However, this could not be further from the truth. Assume for a minute that you are using a homegrown program that uses `dump`. You then add the new XFS filesystem that you just installed to `xfsdump`'s include list. The first thing that `xfsdump` does is *rewind the tape*, whether or not the no-rewind device was chosen. It then attempts to read the first block of data on the tape. Depending on the complexity of the script that called `xfsdump`, the first file on the tape could be an electronic label that the script put on the tape, or it could be the first `dump` backup that went to the tape. In the latter case, `xfsdump` says, "This is not an `xfsdump` backup...I will overwrite it." If it *is* an `xfsdump` backup, `xfsdump` does not overwrite it but appends to it.

Another thing about `xfsdump`, perhaps its most "interesting" feature, is that it writes multiple tape files per

`xfsdump` backup. Typically, each `dump` backup creates one tape file on the tape, but `xfsdump` uses an algorithm to determine how many files it should place on the tape. This supposedly makes recovery quicker, but it also makes it completely incompatible with almost all homegrown shell scripts.

The best thing to do here is be prepared. Know which versions of `dump` and `restore` you use, and experiment with them to see if they can read each other's volumes. If you are talking about two versions of `dump` on the same system, it will probably either always work or never work. Remember to test, test, test.

Syntax of the restore Command

Once you can read a `dump` volume, you need to decide what data needs to be read and how to read it. This section discusses commonly used arguments to `restore` and when to use them.

Essentially, there are four things you might want to do with a `dump` volume:

- Read the table of contents to verify its contents
- Restore an entire filesystem
- Restore selected files
- Perform an "interactive" restore

The first three uses of `restore` can take their data from standard input. These are the appropriate ways to use the command if you must pipe data to them, such as in the preceding `dd` example. The interactive restore works well only when it can see the whole dump file or tape. The syntax of a normal `restore` command is as follows:

```
$ restore [trxi]vbsfy blocking-factor file-number device-name
```

The Options to the restore Command

How `restore` behaves depends on what types of arguments you pass to it.

Determining the type of restore

The first argument to `restore` specifies what *type* of restore to perform. You may specify only one of four possible arguments:

`t`

: Tells `restore` to display a table of contents of the volume

`r`

: Specifies that the entire contents of the volume should be restored to the current working directory

`x`

: Tells `restore` to extract only the files listed at the end of the command

`i`

: Allows you to perform an interactive restore

Determining how the restore behaves

The rest of the arguments are optional and specify how `restore` behaves during the process:

`v`

: Specifies verbose output

`s`

Tells `restore` to skip some number of tape files before it begins reading the tape

`b`

Allows you to specify the blocking factor of the volume you are reading

`f`

Specifies the filename of the backup drive (or disk file) you are using

`y`

Tells `restore` to attempt to recover from read errors

The following sections explain these options in more detail.

Creating a dump volume table of contents (t)

The `t` option is used to see what files are contained on a `dump` volume. This is a good command to include in any automated shell script that controls your `dump` backups. It is also handy on the backend if you are unsure of things such as the case or exact locations of the filenames. You can extract the list of files on any `dump` volume into a file, then use tools like `grep` to find the files you are looking for. For example:

```
# restore tfy device >/tmp/dump.list
```

The preceding command reads the table of contents of the `dump` backup on *`device`*, and sends its output to */tmp/dump.list*. The following command searches */tmp/dump.list* for the phrase *`filename`*:

```
# grep filename /tmp/dump.list
3455            ./somedirectory/filename
```

Performing a complete (recursive) filesystem restore (r)

The `r` option is designed to restore an entire filesystem by reading the entire contents of a `dump` volume into a filesystem. This should be used only if you are absolutely sure that you want to restore the entire filesystem. It requires that you start with the level 0 dump file and then optionally read any incremental backups. It writes the file *restoresymtable* (called restoresmtable on some Unix versions) and references that file when reading the incremental restores. An incremental `dump` records the time of the lower-level `dump` on which it was based. Since the `r` option is designed to restore an entire filesystem, it does not allow you to read an incremental `dump` that is based on a `dump` volume that has not been read yet. For example, suppose that you have three `dump` backups, a level 0 from Monday, a level 1 from Tuesday, and a level 2 from Wednesday. If you read the level 0 using the `r` option and then try to read the level 2 without reading the level 1, `restore` complains.

Tip ✎

You should remove the *restoresymtable* file when the entire restore is complete. (Do not remove it until you have read all levels of your backup tapes, however.)

To use this option, first `cd` into the filesystem that you want to restore, then load the level 0 backup and execute the following command:

```
# restore rbvsfy blocking-factor file-number device-name
```

For example, to restore the entire contents of a `dump` tape that was made with a blocking factor of 32 and is sitting in */dev/rmt/0cbn*, issue the following command:

```
$ restore rvbfy 32 /dev/rmt/0cbn
```

After this command completes, load any incremental backups, starting with the lowest-level backup, and execute the same command again. Do this until you have loaded the most recent incremental backup. If you have more than one `dump` volume of the same level, you need to load only the most recent one. For example, if you make a level 0 once a month and make level 1 backups the rest of the month, to restore the entire filesystem you need to load only the original level and then the latest level 1.

Restoring files by name (x)

You can use the `x` option if you know the exact name and path of the file(s) you want to restore. (Not all `restore` versions that I tested support using wildcards in the include list, so you do need to know the *exact* filenames.) It basically makes `restore` work like `tar`, allowing you to list on the command line the files to be extracted. Keeping in mind that all `dump` backups are made with relative pathnames, you need to `cd` into the filesystem where you want the file(s) to reside. Then, execute the following command to extract the file(s) from the backup:

```
# restore xbvsfy blocking-factor file-number device-name ./dir/file1 ./dir/file2
```

For example, to restore the files */etc/hosts* and */etc/passwd* from a `dump` tape that was made with a blocking factor of 32 and is sitting in */dev/rmt/0cbn*, issue the following command:

```
$ restore xvbfy 32 /dev/rmt/0cbn ./etc/hosts ./etc/passwd
```

Restoring files interactively (i)

This is the option that differentiates `restore` from `tar` and `cpio`. When `dump` makes a backup, it stores at the beginning of the dump an index of what it is about to back up. (As with the other `restore` modes of operation, you

should `cd` into the filesystem where you want the restored files to reside before executing the `restore` command.) The interactive option simulates mounting the `dump` volume and establishes a mock shell where you can use the following commands: `cd`, `ls`, `pwd`, `add`, `delete`, and `extract`. You can use these commands to maneuver around the directories listed on the `dump` volume much as if you were moving around a filesystem.

When you see a file that you want to include in your restore, simply enter `add` *`filename`*. Most versions of `restore` also support shell wildcards here, too, so you can also enter `add` *`*pattern*`*. Once a file is selected for a restore, an asterisk appears next to it the next time you ask for a file listing with `ls`. If you notice that you have added a file that you do not want to restore, just enter `delete` *`filename`* or `delete` *`*pattern*`*. This, of course, does not delete the file from the volume; it merely drops that file from the list of files to be extracted. Once you have selected the files that you want to restore, simply type `extract`.

`restore` then asks a question about which volume to start with. This question is relevant only if you are restoring a few files that are spread across multiple tapes. Because the files are dumped in inode order, you can put the last tape in first, and `restore` can read the first file's inode number and tell immediately if it needs to read anything on that tape; if so, it has to read only up to the last inode on that tape. If it still needs to read files off the other tapes, put them in the drive in decreasing order; again, it knows whether it has to read those tapes and how much of them to read. If you put tape 1 in first, it simply reads the tapes sequentially. If you are restoring a filesystem, this works just fine.

If you are restoring a few files from a `dump` backup that spans multiple tapes, put the tapes in the drive in reverse order and answer with the appropriate number. If you have

only one tape or are just going to read the tapes sequentially, just enter the number 1.

The file(s) that you selected are then restored into the directory where you were when you entered the `restore` command. (`restore` makes any directories that it needs to restore the files.) Once the restore has completed, it asks you, `set owner/mode for '.'`? Many people don't understand what this question means. Assume that you backed up */home/curtis*, which was owned by the user *curtis*. If you are restoring that home directory to */tmp*, answering "Yes" results in the */tmp* being owned by the user *curtis*! Therefore, be careful when restoring files to alternate locations and answering "Yes" to this question. Answering "No" results in the directory permissions being left as they are.

Example 3-1 is a sample `restore` session. Most of the extra verbose comments that you see here, such as block size, the date that `dump` made the volume, and other messages, are the result of adding the verbose (`v`) option (the verbose option is discussed later in this section). In this session, the file */etc/passwd* is selected and restored to */ tmp/etc/passwd*. (That is because I am sitting in the */tmp* directory when I start the restore.)

Example 3-1. Sample restore session

```
# cd /tmp
# ufsrestore ifvy /tmp/dump
Verify volume and initialize maps
Media block size is 126
Dump   date: Sun Apr 30 23:07:22 2006
Dumped from: Sun Apr 30 22:15:37 2006
Level 9 dump of / on apollo:/dev/dsk/c0t0d0s0
Label: none
Extract directories from tape
Initialize symbol table.
ufsrestore > ls
.:
     2 *./            2 *../        11395  devices/   28480
```

```
etc/

ufsrestore > cd etc
ufsrestore > ls
./etc:
 28480  ./              2 *../         28562  dumpdates   28486
passwd

ufsrestore > add passwd
Make node ./etc
ufsrestore > ls
./etc:
 28480 *./              2 *../         28562  dumpdates   28486
*passwd

ufsrestore > extract
Extract requested files
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
extract file ./etc/passwd
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
ufsrestore > q
# ls -lt /tmp/etc/passwd
-rw-r--r--   1 root       sys       34983 Apr 28 23:54
/tmp/etc/passwd
```

Restoring files to another location

All filenames on a dump backup volume have a relative pathname. In other words, if you back up */home*, which includes */home/mickey* and */home/mouse*, the listing looks like this:

```
15643   ./mickey
12456   ./mouse
```

So, restoring the files to an alternate location is very easy. Simply change directories to something other than the original mount point (e.g., */home1*) and start the restore from there. restore creates directories as needed. If you change

the directory */home* to */tmp* in the preceding example, it creates */tmp/mickey* and */tmp/mouse*.

Requesting verbose output (v)

The `v` option does not require an argument and results in a verbose output. It displays a lot of extra information, such as the date and level of the backup, as well as the name of each file as it is restored.

Tip ✎

The `s`, `b`, and `f` options require an argument. These options work just like their counterparts in the `dump` command. (This is not to say that the `s` option performs the same function in both commands, though.) List all the options you want to use just after the `restore` command, then list each option's accompanying argument in the same order as you listed the options. For example, to use the `b`, `f`, and `s` options, issue the following command:

```
# restore tbfsy blocking-factor device-file file-number
```

Skipping files (s)

The `v` option is used to read a `dump` backup other than the first one on a tape. When you issue multiple `dump` commands to a nonrewinding tape device, each becomes a separate file; files are separated by an EOF mark. You cannot read all of these in one stroke with a single command. (If you were restoring, you probably wouldn't want to, because each is probably a backup of a separate

filesystem.) You have to read each backup with a separate `restore` command. There are two scenarios here. You can:

- Consecutively read every filesystem from the tape, such as when you want a table of contents of the entire tape.
- Read a certain filesystem from a tape.

Reading multiple filesystems consecutively may be accomplished by simply executing several `restore` commands in a sequence, using the nonrewinding tape device. Whether this works for you depends on how your system's tape device driver functions. After a successful execution of a `restore` command, the tape may stop at the end of the file just after the EOF mark. If it is a Berkeley-style device, it may stop at the end of the file just *before* the EOF mark. In that case, the next `restore` command would fail. You sometimes can fix this by executing one forward space file command (e.g., `mt -t` *`device`* `fsf 1`). This positions the tape just after the EOF mark, and you can then execute your next `restore` command.

Reading a certain filesystem's `dump` backup from tape can be accomplished one of two ways. You can:

- Position the tape to the appropriate dump file using `mt` or `tctl` and then execute your `restore` command with no `s` argument.
- Rewind the tape and use the `s` option to tell `restore` which file to read. It then forwards the tape to that file and reads it. `s` requires an argument, from 1 to *`n`*. This value should be the number of the file that you want to read from the tape. The first backup on the tape is numbered 1, so issuing the command `restore tsf 1` *`device`* is functionally the same as `restore tf` *`device`*.

Tip ✎

Please note the difference between `mt` and `restore`. The way `mt` and `restore` number the tape files is off by one. If you want to tell `mt` to go to the second file on tape, issue the command `mt -t` *`device`* `fsf 1`. If you want `restore` to read the second dump volume on the tape, issue the command `restore [irtx]s 2`. This has confused more than one system administrator!

Specifying a blocking factor (b)

The `b` option explicitly tells `restore` what blocking factor `dump` used when writing the volume. It requires an argument that is a numeric value, normally between 1 and 126, or the highest blocking factor that your version of `dump` supports. This blocking factor is multiplied by the minimum block size that your version of `dump` supports. The minimum block size is usually 1,024 but may be 512. (Check your version's manpages.) Many versions of `restore` can now automatically detect most common blocking factors, so you may not even need this option. If you determine that you have a blocking factor that your version of `restore` cannot automatically detect, use it to tell `restore` which blocking factor was used. If you are using `dd` to read the data and pipe it into `restore`, you do not need to use the `b` option.

Specifying a backup drive or file (f)

The `f` option is used quite often, and it tells `restore` to read from the device specified in the accompanying argument, instead of the default tape drive for your version of Unix. The argument may specify any of the following:

`/dev/rmt/0`

A local device name (e.g., */dev/rmt0*, */dev/rmt/8500compressed)*

`/backup/dumpfile`

Any backup file that was created by `dump`

`remote_host:` `/dev/rmt/0`

A remote device, by specifying a hostname prior to the device (Not all versions of restore support the use of remote hosts.)

> **Tip ✎**
>
> Be sure to read "Using ssh or rsh as a Conduit Between Systems" near the end of this chapter for a more secure way to use remote devices.

`"-"`

Standard input, such as when reading from `dd`, or a `dump` sent to standard output

Specifying no query during restore (y)

Normally, when `restore` encounters an error in the file, it stops and asks you if you want to continue. If you add the `y` option, it does not ask you this question and tries to continue as best it can when it encounters an error.

Limitations of dump and restore

`dump` and `restore` have many capabilities. A good shell script can automate their use and can provide a very good safety net for that time when your disk goes south. However, these utilities do have their limitations:

- There is no way with `dump` to get a consistent picture of an entire filesystem at any given moment in time.
- The `dump` command is sometimes silent about open files and other problems, although it complains with a "bread error" if things get really confused.
- When files are skipped, `restore` can actually make you think they are on the volume.
- You do need to write scripts to work with `dump`, and scripts can have errors.
- There are multiple versions of `dump`, not all of which play well with one another.
- Like all native utilities, `dump` and `tar` lack online indexes like those available with commercial utilities. (Solaris's version of `dump` does have an a option that performs some level of indexing, but it definitely isn't the same as what you'd get with a commercial product.)

As long as you keep these issues in mind, you can get by for a long time using `dump` and `restore` and avoid spending anything extra for commercial software. Have fun!

Features to Check For

If you're going to write your own script to work with `dump` or any other commands in this chapter, make sure that whatever backup script you use does the following:

Lots of error checking

> I have seen too many shell scripts over the years that assume things. Do not assume that a simple command worked just because it always does. When you are automating things, check the return code of *everything*. If you can anticipate what causes a given error, try writing the script so that it fixes that error before you completely give up.

Notification, notification, notification

> I cannot emphasize this enough. If your script sees something that it isn't used to seeing, you should be notified. All good activities should also be logged so that you can check those logs to make sure everything worked. Too many restores have failed because someone didn't read her backup log. If you do have a script that notifies you when things go wrong, don't assume that nothing is wrong if you don't get mail. What if `cron` is down? What if some minor change that you made to the script causes it to abort without a notification? What if `sendmail` was or is down? *Never assume anything.*

Proper checking of an rsh or ssh command

> Too many scripts check the return code of the `rsh`/`ssh` command and not the return code of the command that was executed on the remote machine. Try this sometime: issue one of these commands:
>
> ```
> $ rsh remote-system do_stuff ; echo $?
> $ssh remote-system do_stuff ; echo $?
> ```

where *remote-system* is a system that you can `rsh` or `ssh` to, and *do_stuff* is a command that does not exist on that system. You will see that the command that you issue fails on *remote-system*, but `ssh/rsh` returns a successful return code of 0. That is because the `rsh/ssh` command succeeded, whether the command it issued succeeded or not. That is why you need syntax such as the following (`ssh` works here as well):

```
rsh apollo "ls -l /tmp/* ; echo \$?>/tmp/ls.success"
SUCCESS=\Qrsh apollo cat /tmp/ls.success ; rm
/tmp/ls.success\Q
if [ $SUCCESS -eq 0 ] ; then
    #everything worked
    echo "Everything worked."
else
    echo "Something bad happened!"
fi
```

This shows you the return code of the remote command, instead of just that of the `rsh` or `ssh` command.

The preceding syntax does not work with `csh`, because it does not allow output redirection in the same way. One way to get around the `csh` problem is to create a small script that you `rcp` over. That script can explicitly call `/bin/sh`, so you can be sure you are getting that shell.

Get the table of contents from the backup volume

You always should reread your backup volumes, for two reasons. The first is that it is the best verification that the backup worked, short of actually restoring the data. The second is that you can store these tables of contents into a file and use that file during an actual restore to find out which volume has the file you are looking for.

The best way to verify that the `dump` volume is intact is to list the table of contents with the verbose option turned on, sort by inode number, and restore the last

file. This reads the whole volume and ensures that the dump is intact all the way to the last file.

Backing Up and Restoring with the cpio Utility

`cpio` is a powerful utility. Unlike `dump`, it works on the file level. For this reason, it handles changing filesystems a little better than `dump`, but it changes the access time (`atime`) of files as it is backing them up. (It does have an option to reset `atime`, but this changes `ctime`.) Unless you're using GNU `cpio`, one of `cpio`'s biggest challenges is compatibility between different operating systems. In addition, `cpio` requires you to specify files to include on standard input, which makes it a bit different from all other backup tools.

`cpio` does make you do more work than `dump` does. This means you need to know a little bit more about how it works if you want to use it for regular system backups. You need to understand:

- How to use `find` with `cpio` to do full and incremental backups of a filesystem, while leaving the access time (`atime`) of the files unmodified
- What arguments give you the best results
- How to use `rsh` or `ssh` to send a `cpio` backup to a remote backup drive
- How to get a table of contents of that volume
- How to manipulate a tape drive and restore from a backup created by `cpio`

One good thing about `cpio` is that its name is usually `cpio`. (A great advantage over `dump` to be sure!)

> **Tip ✎**
>
> Mac OS users: Remember to use the native `cpio` if you're running a version of Mac OS later than 10.4. Otherwise, use `ditto` if you need `cpio` format.

Let's start with the basic syntax of `cpio`, followed by some example commands.

`cpio`'s backup syntax is as follows:

```
cpio -o [aBcv]
```

`cpio`'s restore syntax is as follows:

```
cpio -i [Btv] [patterns]
```

The following example command creates a full backup of */home* to a local tape drive:

```
$ cd /home
$touch level.0.cpio.timestamp
```

The `touch` command is optional, but it makes incremental backups possible.

```
$ find . -print|cpio -oacvB > device
```

Of course, the device in the preceding command also could be a local file if you are backing up to an optical or CD device. This command creates an incremental backup of */home* to a local tape drive:

```
$ cd /home
$ touch level.1.cpio.timestamp
$ find . -newer level.0.cpio.timestamp -print \
  |cpio -oacvB > device
```

These commands create a full backup of */home* to a remote tape drive:

```
$ cd /home
$ find . -print|cpio -oacvB \  |(rsh remote_system dd of=device bs=5120)
```

Here's a more secure method that uses `ssh`:

```
$ find . -print|cpio -oacvB \  |(ssh remote_system dd of=device bs=5120)
```

The Syntax of cpio When Backing Up

The `cpio` command takes its list of files from standard input (`stdin`) and by default sends its data stream to standard output (`stdout`). To provide a list of files to back up, do anything that generates a list of files:

- Use `ls` or `find` (e.g., `ls | cpio -oacvB`).
- Create an include file, then send it to the `stdin` of `cpio` (e.g., `cat /tmp/include | cpio -oacvB`, or `cpio -oacvB </tmp/include`).

All the preceding references generate an include list with a path that is *relative* to the current working directory. This is done automatically with `dump`, but with `cpio`, you can use either relative paths (e.g., `cd /home;find .`) or absolute paths (e.g., `find / home1`). However, using absolute paths severely limits your restore flexibility. If a table of contents of your `cpio` file shows */home1/directory/somefile*, you can restore it only to */ home1/directory/somefile*. (Sometimes it is possible to use `chroot` to fix this, but it is very tricky!) On the other hand, if the table of contents shows *./home1/directory/somefile* or *home1/directory/somefile*, you can restore it to anywhere you want by changing to another directory and running the restore from there. Therefore, you should always use relative paths when creating include lists for `cpio` or `tar`. (GNU `tar` suppresses absolute paths during a restore, but it is probably better to develop a habit of using relative paths when creating include lists for either of these backup utilities.)

`find` is the usual method for making regular system backups because it can make `cpio` perform incremental backups. Before beginning a full backup of a filesystem or directory, create a timestamp file in the top-level directory. For example, in the native version of `cpio`, if you want to do incremental backups of */home1*, create a file called */home1/level.0.cpio.timestamp*. Then perform the full backup, using a `find` command that lists the entire contents of that directory or filesystem (e.g., `find . -print`). When it is time for a level 1 backup, you create the file */home1/level.1.cpio.timestamp* and use a `find` command that looks for files newer than */home1/level.0.cpio.timestamp* (e.g., `find . -newer level.0.cpio.timestamp`). The *level.1.cpio.timestamp* file can then do a level 2 backup, using a `find` command that looks for files newer than that file. You can use this technique to generate as many levels of backups as you wish.

The Options to the cpio Command

There are six options that should be used when making regular `cpio` backups. The first five usually are listed all at once (e.g., `-oacvB`), and the last one usually is listed as a separate argument (e.g., `-C 5120`). (Note that the `-B` and `-C` options are mutually exclusive; they cannot be used together.)

`o`

The `o` option specifies that a backup should be created.

`a`

The `a` option resets `atime` to its value before the backup.

`c`

The `c` option tells `cpio` to use the ASCII header format.

`v`

The `v` option results in verbose output.

`B`, `C`

The `B` and `C` options let you specify the block size.

In addition, you can specify a device or file to which `cpio` can send its output rather than sending it to `stdout`. All of these options and more are available in the GNU version of `cpio`, as is the ability to use remote devices.

Use GNU cpio if You Can!

GNU `cpio` brings a lot of functionality to the table, and there are three very good reasons for using it if you can:

The native `cpio` utility is not very portable, even when it says it is. However, if you write a backup using GNU `cpio`, you can always read it as long as you have GNU `cpio` on your system—no matter what platform it is.

The portable ASCII format also has limitations. For example, it cannot handle a filesystem with more than 65,536 inodes. The `newc` header format available in GNU `cpio` has overcome this limitation.

It supports remote devices just like `dump`! As long as it's OK to use `rsh` authentication, all you have to do is enter:

```
$ -O remote_host:/device_name
```

GNU `cpio` is available at http://www.gnu.org.

Specifying the output mode (o)

The `o` option is one of the three modes of `cpio` (`o`, `i`, and `p`) and is used to create a backup. It is listed as the first of several arguments.

Restoring access times (a)

One of the differences between `dump` and `cpio` is that `dump` backs up directly using the disk device, whereas `cpio` must go through the filesystem. Therefore, when `cpio` reads a file to back it up, it changes its access time (`atime`). System administrators typically use this value to see when a user has last used a file by looking at it in some way. Files that have not been accessed in a long time are typically removed from the system as part of a cleanup process. If your backup program changes the access time of a file, it appears as if all files are used every night. This option to `cpio` can reset `atime` to its original value.

> **Warning ☠**
>
> Restoring access times causes `ctime` to change. This could trigger some hacker alerts if you're watching these things closely.

Specifying the ASCII format (c)

When `cpio` backs up, it can send the data to the backup device using a number of header formats. These formats can be very platform-dependent, and therefore not very exchangeable between systems. The most exchangeable format (although not completely exchangeable) is called the ASCII format. The `c` option tells `cpio` to use this format. As mentioned in the sidebar "Use GNU cpio if You Can!", this format may not be as interchangeable as you might think. If you are really concerned with portability, you should consider using GNU `cpio`. If you can't use it, you should try transferring `cpio` files between the different

flavors of Unix that you have. At least you will know where you stand. Either way, using the `c` option can't hurt.

Requesting verbose output (v)

The `v` option causes `cpio` to print the list of files that it backs up to standard error (`stderr`). The actual data of the `cpio` backup goes to standard out (`stdout`). (The backup data always goes to `stdout`, unless your version of `cpio` supports the `-O` option, which can specify an output file or device.)

Specifying a blocking factor of 5,120 (B)

The `B` option simply tells `cpio` to send its data to `stdout` in blocks of 5,120, instead of the default block size of 512. This can help the backup to go faster. However, it is nowhere near the large blocking factors that many modern backup drives prefer. You should therefore use the `C` option listed next if it is available on your system. The two options are mutually exclusive.

Specifying an I/O block size (C)

The `C` option does require an argument and allows you to specify the actual block size. If you are on AIX, the value is a blocking *factor*, which is multiplied by the minimum block size of 512. Most other Unix versions allow you to specify the value in bytes.[3]

Either way, you can set this value to be quite large, allowing `cpio` to perform much better with modern backup drives. Once again, this option is mutually exclusive with the `B` option and usually is listed separately with its argument, as in the following example:

```
$ find . -print|cpio -oacv -C 129024 >device
```

Specifying an output device or file (O)

Some versions of `cpio` allow you to specify a `-O` *`device`* argument, which causes the output to go to *`device`*. (This option is not always available.) All versions of `cpio`, however, default to sending the backup data to `stdout`. Once again, for simplicity, you don't have to use the `-O` option even if it is available. To specify a backup device, simply redirect `stdout` to a file or device. This method always works, no matter what version of Unix you are using.

Backing up to a remote device (piping to an rsh or ssh command)

The native version of `cpio` does not automatically support remote devices in the way that `dump` does. (The GNU `cpio` version does do this.) So, in order to back up to a remote backup drive, you need to replace the `>` *`device`* option with a pipe to an `rsh` or `ssh` command:

```
$ find . -print|cpio -oacv \
| rsh remote_system dd of=devicebs=5k
```

Here's a more secure version:

```
$ find . -print|cpio -oacv \
| ssh remote_system dd of=devicebs=5k
```

Notice that it is piped to a `dd` command on the remote host. Since the input file is `stdin`, you need only specify the output file (`of=`) and the block size. You need to specify the 5 K block size because that is readable by any version of `cpio`.

Restoring with cpio

The same rules apply to `cpio` as to any other `restore` command. I hope that you aren't sitting there with a `cpio` volume in your hand that contains your very critical system backup, and you've never restored with `cpio` before. Remember, test, test, test, and practice, practice, practice! OK, now that I'm off my soapbox, don't worry. Restoring from a `cpio` volume isn't that hard, although there are a number of possible challenges that you may face when trying to read a `cpio` volume.

> **Tip ✎**
>
> This next section assumes that you know the volume was made with `cpio` and that you know its block size. If you do not have this information, see the section "How Do I Read This Volume?" in Chapter 23.

Different versions of cpio

Just because you know that a backup volume was written in `cpio` format doesn't mean you can read it easily. This is because, although most versions of `cpio` are *called* `cpio`, they don't always produce the same format. Even the ASCII header that is intended to provide portability is not readable among all platforms. If you just want to see if you can read the volume, try a simple `cpio -itv <` *`device`*. If that works, then you're golden! If it doesn't work, you might get errors like:

```
Not a cpio file, bad header
```

or:

```
Impossible header type
```

Tip ✎

GNU `cpio` can save you hours of work. If you have GNU `cpio`, you could skip this whole section. The following is an excerpt from the GNU `cpio` manpage: “By default, `cpio` creates binary format archives, for compatibility with older `cpio` programs. When extracting from archives, `cpio` automatically recognizes which kind of archive it is reading and can read archives created on machines with a different byte-order.”

Byte-order problems

If you are reading the volume on a type of platform that is different from the one on which the volume was written, you might have a byte-order problem, and you will probably get the first of the two preceding errors. The `b`, `s`, and `S` options to `cpio` are designed to help with byte-order problems:

```
$ cpio -itbv < device
# Reverse the order of the bytes within each word.
$ cpio -itsv < device
# Reverse the order of the bytes within each half word.
$ cpio -itSv < device
# Swap half word within each word
```

Warning ☠

Reversing the byte order may allow you to read the `cpio` header, but it may render the restored files useless. If the volume was not made with the `c` option, your best bet is to restore it on a system with the same byte order. (Consult the section "How Do I Read This Volume?" in Chapter 23 for more information about byte order.)

Wrong header type

If you don't have a byte-order problem, the `cpio` data might have been written with a different type of header. Some versions of `cpio` can automatically detect some of the headers, but they can't detect all of them, and some versions of `cpio` can detect only one type automatically. You may have to experiment with different headers to see which one it was written in. If this is your problem, you are probably getting the "Impossible header type" error. (Again, GNU `cpio` is able to detect any header type automatically.) Try some of the following commands:

```
$ cpio -ictv <device
# Try reading the incoming data in ASCII format
$ cpio -itv -H header <device
# Try reading with a header of value header
```

The value *header* could be `crc`, `tar`, `ustar`, `odc`, and so on. Consult your manpage. This option is not available everywhere.

```
$ cpio -ictv -H header <device
# Combining ASCII and header options
```

Strange block size

Finally, the `cpio` volume could have been written with a block size other than what `cpio` expects. If the block size of your `cpio` backup is 5 K, you can try telling `cpio` to use that block size by adding the `B` option to any of the preceding commands (`cpio - itBv`). If the block size is not 5 K, you can get `cpio` to use it by adding a `-C` *`blocksize`* at the end of the `cpio` command (`cpio -itv -C 5120`).

Full or partial restore, or table of contents only?

Once you determine that you can read the `cpio` backup volume, you have several choices of what to do with it:

- Restore the contents into the current directory or filesystem.
- Restore files that match the pattern you specify. This "pattern" can be the ouput of a command.
- Do either of the preceding while interactively renaming the files.
- Read the table of contents.

cpio's Restore Options

Before doing any of the things just described, you have several options available to read from a `cpio` volume. Many of these are the same options that you used to create a `cpio` volume, such as (`B`) for 5 K blocks, (`c`) to read an ASCII header, and (`v`) to give verbose output. In addition, you have the following:

`i`

The `i` option starts out the restore options string and tells `cpio` that it is in input mode.

`t`

If the `i` option is followed by a `t`, `cpio` generates a table of contents. It does not actually restore anything from the volume.

`k`

The `k` option tells `cpio` to attempt to skip bad spots in the volume.[4]

`d`

The `d` option causes `cpio` to make directories as needed.

`m`

The `m` option tells `cpio` to restore the original modification times of the files when they were backed up. Otherwise, `cpio`'s default action is that the modification times of a restored file are set to the time of the restore.

> **Tip ✎**
>
> Note that `cpio`'s default action in this regard is the opposite of `tar`'s default action.

`u`

This option tells `cpio` to unconditionally overwrite all files.

`"*` *`pattern*`* `"`

This option restores files that match the pattern.

`f "*` *`pattern*`* `"`

This option restores files except those that match the pattern.

`r`

This option tells `cpio` to interactively rename files. If any files are restored, the user is asked to rename each file as it is restored. If the user enters a null value, the file is not restored.

Telling cpio Which Device to Use

Unlike `tar` or `dump`, `cpio` does not take the name of the backup device as an argument.[5]

You must feed `cpio` the data through `stdin`. You can do this the hard way by using `dd` or `cat`:

```
$ dd if=device bs=blocksize| cpio -options
```

Alternatively, you can simply redirect `stdin` to read from the device:

```
$ cpio -options< device
```

Examples of a cpio Restore

The only question now is what options are needed. The easiest way to explain this is to show you example commands for the things that you can do with a `cpio` volume. Several “optional” options are listed in these example commands. Many of these options, while not required, make the operation easier or more robust. Some of the options may not be applicable to your particular application, so feel free to not use them.

Listing the files on a cpio volume

The following command reads the `cpio` volume in (`B`) blocks of 5120 bytes, uses the (`c`) ASCII format when reading the header, (`k`) skips bad spots on the volume when possible, and lists only the (`t`) table of contents with a (`v`) verbose (`ls -l`) style listing:

```
$ cpio -iBcktv <device
```

Doing an entire filesystem restore

The following command reads the `cpio` volume in (`B`) blocks of 5,120 bytes, uses the (`c`) ASCII format when reading the header, and makes (`d`) directories where needed. It (`k`) skips bad spots on the volume when possible, retains the original file (`m`) modification times, (`u`) unconditionally overwrites files, and (`v`) lists the names of the files that it recovers as it reads them:

```
$ cpio -iBcdkmuv <device
```

Of course, you can do the same thing, but without the (`u`) unconditional overwrite:

```
$ cpio -iBcdkmv <device
```

Doing a pattern-match restore

To restore files that match a certain pattern, simply list the pattern(s) you are looking for after the command:

```
$ cpio -iBcdkmuv "pattern1" "pattern2" "pattern3" < device
```

The *`pattern`* uses filename expansion wildcards, not regular expressions.[6]

Filename expansion wildcards work like the ones on the command line (e.g., `*ome*` finds both *home1* and *rome*). The `cpio` command is the only native restore utility that supports wildcard restores in this way. For example, if you

want to restore all of the files that were in my home directory (*/home1/curtis*), you can type:

```
$ cpio -iBcdkmuv "*curtis*"
```

> **Warning ☠**
>
> Quoting the pattern as shown in the previous code causes the filename expansion to be applied to the files in the archive. If you don't quote the pattern, the shell expands the wildcard for you, and `cpio` sees a list of filenames that currently exist on the system and match the pattern `*curtis*`. If you have deleted some of these files or if you are in a different directory, the results will not be what you expect!

To restore all files except those matching a certain pattern, use the `f` option, and list the excluded pattern(s):

```
$ cpio -iBcfdkmuv "pattern1" "pattern2" "pattern3" <device
```

Renaming files interactively

The following is the same command as that in the previous section "Doing an entire filesystem restore" but prompts the user to interactively (`r`) rename any files that are restored:

```
$ cpio -iBcdkmruv < device
```

The following is the same command as that in the previous section "Doing a pattern-match restore" but prompts the user to interactively (`r`) rename any files that are restored:

```
$ cpio -iBcdkmruv "pattern" < device
```

Other useful options

b, s, S

These options are used to swap bytes when you have byte-order problems. Use them as a last resort, because I've yet to see them used with unqualified success. There is one scenario in which they might come in handy: if you are trying to read a volume that was made on a little-endian machine, but you're on a big-endian machine. (See the section "How Do I Read This Volume?" in Chapter 23 for more information.) The person making the `cpio` backup did not use the `-c` option, so the only way that you can read the volume is to perform a byte swap:

```
$ dd if=device bs=10240 conv=swab | cpio -options
```

Afterwards you discover that the words in the backup are now reversed from the order in which you need them, resulting in restored files that can't be read. Allegedly, you could have `cpio` swap the words for you as they are restored. Notice the addition of the `b` option to the regular `cpio` command:

```
$ dd if=device bs=10240 conv=swab | cpio - iBcdkmubv <device
```

The `b` option is equivalent to using both the `s` and `S` options together. The problem here is that all this byte-swapping is going on without `dd` or `cpio` knowing what the format of the file is. What if the expected 8-byte words aren't 8 bytes at all? What if they're 10? Again, I have not met anyone who has used these options with complete success, so if you do, send me an email!

6

The `6` option reads a Unix sixth-edition archive. Use it for reading *really old* `cpio` backups.

Restoring to a different directory

If you made your backup volumes using relative pathnames, this is not a problem. Simply `cd` to the directory where you want to restore, and issue your `cpio` restore commands from there. If you don't know whether the volume was written with relative pathnames, enter the command `cpio -itv < device`, and look at the filenames. If they start with a /, the volume was made with absolute paths. In that case, you can do one of two things:

Use a symbolic link

> If you are on Unix, the `chroot` command should be available. If you are on a non-Unix platform or the `chroot` command is not available, you may have to be more creative. If you have to restore to a different directory, and the backup was made with absolute pathnames, you might create a symbolic link from */home2* to */home1* (e.g., `ln -s /home2 /home1`). That way, any files that are supposed to go into */home1* actually go into */home2*. This works only if */home1* is not mounted on that system. If */home1* is already present; you must `unmount` it. This, of course, is a pain, which is why you should be making your backup volumes with relative pathnames.

Use GNU cpio

> This is really the best option. GNU `cpio` has a no-absolute-pathnames option that removes the leading slash (/) from any absolute paths and restores the files relative to the current directory.

Using cpio's Directory Copy Feature

If you need to move a directory from one place to another, you can try this little-used feature of `cpio`. Issue the following command:

```
$ cd old-directory ; find . -print | cpio -padlmuv new-directory
```

This moves *old-directory* to *new-directory*, resetting (a) access times, creating (d) directories when needed, (l) linking files when possible, retaining the original (m) modification times, and (u) unconditionally overwriting all files, while giving a (v) verbose output of the files that get copied.

> **Warning ☠**
>
> Some versions of Unix also have a -L option that causes cpio to follow symbolic links, copying the directories and files to which they point, instead of the symbolic link itself. If you use this option, make sure that the find command that is feeding cpio its file list uses the -follow option. If you do not, you will get unpredictable results.

If you were to compile a list of all the options that are available on all Unix platforms, it would be very long. Depending on your platform, there may be a lot of other neat options that can make cpio more useful for you. There are also a number of extra features in GNU's version of cpio. Make sure you read the manpage for your version of cpio. Please be aware that if you use any of the options that affect how the cpio backup is written, it may reduce its portability.

---

[3] This time, it's HP that's the strange one! It doesn't have a similar method for setting block size, and the -C option on HP does something totally different, causing it to use checkpoints. It has nothing to do with the blocking factor at all. (The feature isn't such a bad idea, but couldn't they have used another letter?)

[4] This option is also in GNU `cpio` for compatibility reasons with legacy shell scripts but is actually ignored. GNU `cpio` always attempts to skip bad spots on the tape. Therefore, if you are using `gcpio`, you can drop this option. Some other versions do not have the option at all.

[5] That is, unless you want to use the `-I` option supported by some versions of `cpio`. Once again, though, this book concentrates on those options that work almost everywhere.

[6] For learning more than you ever thought possible about regular expressions, I highly recommend *Mastering Regular Expressions*, by Jeffrey Friedl (O'Reilly). Understanding what they are and what they do is an eye-opening experience and will make your use of tools such as `grep`, `sed`, `awk`, and `vi` much more fruitful.

Backing Up and Restoring with the tar Utility

`tar` is the most popular backup utility discussed in this chapter. Many of the files that you download from the Internet are in `tar` or compressed `tar` format. One limitation of `tar` to consider is that it has always had trouble with exceptionally long pathnames. Although it isn't typically used by itself for daily backup and recovery, GNU `tar` is often used by other open-source tools, such as Amanda (see Chapter 4).

> **Tip ✎**
>
> As mentioned earlier, the native version of `tar` cannot preserve the access times of files that it backs up. If this is important to you, use the GNU version of `tar`; it can do this.

The Syntax of tar When Backing Up

The basic `tar` command is as follows:

```
$ tar [cx]vf device pattern
```

Now let's look at some example commands. To create an archive of a directory called *`pattern`*, use the command:

```
$ tar cvf device pattern
```

To do the same thing but with a blocking factor of 20, use the command:

```
$ tar cvbf 20 device pattern
```

To do the same thing but have `tar` verify the data as it writes it (available only in GNU `tar`),[7] use the command:

```
$ gtar cvWbf 20 device pattern
```

To create an archive of everything in the current directory starting with an “a”, use the command:

```
$ tar cvf devicea*
```

> **Tip ✎**
>
> Remember to use the native Mac OS `tar` if you’re running a version later than 10.4. Prior to that, you’ll need `hfstar`.

The Options to the tar Command

`tar` has two great advantages. The first is the level of acceptance that it has received. The second is its short list of options; there really are not very many:

`c`

The `c` option tells `tar` to *create* an archive (to make a backup).

`v`

The `v` option tells `tar` to be *verbose*. It lists the name and size of each file as it is being archived.

`W`

The `W` option, available only in GNU `tar`, tells `tar` to attempt to verify the files as it writes them.

`b` *`blocking-factor`*

This option tells `tar` to read and write in blocks of *`n`* bytes, where *`n`* is the value of the *`blocking-factor`* (that you specify) multiplied by the minimum block size (for that operating system). This is normally 512 but could

be 1,024. The resulting value, referred to as the *block size*, can range from 512 to 10,240. A block size of 10,240 would normally mean a blocking *factor* of 20, because 20 times 512 is 10,240. There is a default value for `b` if you do not specify it. This default value is usually 20 but could be as little as 1.

`f` *`device`*

This option tells `tar` to write to the device specified in the *`device`* argument, instead of the default tape device for that platform. This *`device`* could be a file on disk or optical platter, a tape drive, or standard output (`stdout`). If you are using GNU `tar`, it also could be a remote system's tape drive (see the following sidebar "Use GNU tar if You Can"). To send the data to `stdout`, enter a dash (`-`) where the device name should be. (Using `-` is not available on all platforms.)

`pattern`

This is what generates the include list for `tar`. Again, it is based on filename expansion syntax, so to back up everything starting with an "a", you enter "a*" as that argument. You can put any filename here, including a directory; this causes everything in that directory to be archived.

Use GNU tar if You Can

GNU `tar` is an extremely popular utility. Besides being able to read an archive written by any other version of `tar`, it adds a significant level of functionality. Here are some of its most popular advancements:

The `-d` option performs a `diff` compare between the archive and a filesystem. It does this by reading the tape and comparing its contents against the files that it finds in the filesystem. Any differences are reported.

The `-a` option resets access times (`atime`).

The `-F` option runs a script when `tar` reaches the end of a volume. This can be used to automatically swap volumes with a media changer.

The `-Z` and `-z` options automatically pass the archive through `compress` or `gzip`, respectively.

The `-f` option supports remote device names.

By default, GNU `tar` suppresses a leading slash on absolute pathnames while creating or reading a `tar` archive. (You can suppress this with the `-p` option.)

Some people also prefer the GNU style of arguments that are offered by GNU `tar`. Instead of `tar cvf`, you can specify `tar –create –verbose –file`.

GNU `tar` is available at http://www.gnu.org.

Warning ☠

While GNU `tar` can read an archive created by any other version of `tar`, the reverse is not necessarily true. Certain native versions of `tar` cannot read archives created with GNU `tar`.

Listing files on standard input

Most versions of `tar` do not support listing the files to be archived on standard input, like `cpio` does. However, GNU `tar` added this functionality with a –T flag that allows you to specify a file that contains a list of files to be backed up. If you want to specify the names of the files to be backed up via standard input, use GNU `tar` and specify `-` as the include file. This usually tells it to look at standard input

instead of a named file. For example, suppose you wanted to run a `find` from */home/curtis* and back up all the files that you find there:

```
# cd /home/curtis ; find . -print |tar cvf /dev/rmt/0cbn –T -
```

This causes `tar` to see the result of the `find` operation as the list of files to be included.

Some of the native versions of `tar` that support this feature are listed in Table 3-2.

Table 3-2. Versions of tar that support an include list

tar version	Flag
AIX	`-L`
DG-UX, SunOS, Solaris	`-I`
FreeBSD, Linux, GNU `tar`	`-T`

Syntax of tar When Restoring

A `tar` backup is very easy to read. Even if you used a blocking factor when you created the `tar`, you don't need it for the restore. `tar` automatically figures it out. (Did I hear you say "How beautiful..."?) To read a backup written with `tar`, enter:

```
$ tar xvfdevice
```

or:

```
$ tar xvf device pattern
```

The `x` flag tells it that you are extracting (restoring) from the `tar` file. The `v`, `f`, and *`device`* arguments work the same way as they do when making a backup.

Restoring selected parts of the archive

When restoring, you can specify the filename(s) that you want to restore by listing one or more *pathnames* after the device name. It is important to note, however, that the pathname must match the name in the `tar` archive *exactly*, or it is not restored. Unlike in `cpio`, wildcards are not supported in `tar`. However, if you specify a directory name, everything in that directory is restored. Remember, your specification must match the directory name exactly.

Consider the following example. There is a subdirectory called *home*, and we create a `tar` archive of it, called *file.tar*. You can enter `tar cvf file.tar home` or `tar cvf file.tar ./home`. Watch how that affects what you must do to restore from it:

```
$ tar cvf home.tar ./home
a ./home/ 0K
a ./home/myfile 0K
a ./home/myfile.2 0K
```

If it was backed up with *./home*, it must be restored with *./home*:

```
$ tar xvf home.tar home
tar: blocksize = 5
$ tar xvf home.tar ./home
tar: blocksize = 5

x ./home, 0 bytes, 0 tape blocks
x ./home/myfile, 0 bytes, 0 tape blocks
x ./home/myfile.2, 0 bytes, 0 tape blocks
```

This time it is backed up with *home* as the pattern:

```
$ tar cvf home.tar home
a home/ 0K
a home/myfile 0K
a home/myfile.2 0K
```

Notice again that if it was backed up with *home*, it must be restored with *home*. The pattern of *./home* does not work:

```
$ tar xvf home.tar ./home
tar: blocksize = 5
```

```
$ tar xvf home.tar home
tar: blocksize = 5
x home, 0 bytes, 0 tape blocks
x home/myfile, 0 bytes, 0 tape blocks
x home/myfile.2, 0 bytes, 0 tape blocks
```

If you don't know the name of the file you want to restore and you don't want to restore the entire archive, you can create a table of contents and look for the file there. First, make a table of contents of the archive:

```
tar tf device> somefile
```

If you do that with the archive in the preceding example, you will have a file that looks like this:

```
home/
home/myfile
home/myfile.2
```

If you knew you were looking for *myfile*, you could `grep` for that out of this file:

```
# grep myfile somefile
home/myfile
home/myfile.2
```

You would then know that you should enter:

```
$ tar xvf device home/myfile
```

Tricking tar into using wildcards during a restore

There is a trick that works most of the time on tape and should work all of the time for `tar` files on disk. Issue two `tar` commands at once:

```
$ tar xvf device \Qtar tf device | grep 'pattern'\Q
```

If you are using this trick with a tape drive, make sure you use the rewind device, or it won't work! You also might want to add the `sleep` command to give the tape time to rewind:

```
$ tar xvf device \Qtar tf device | grep 'pattern' ; sleep 60\Q
```

Changing ownership, permissions, and attributes during a restore

The default actions of `tar` can vary from system to system, but most versions of `tar` support the following three options during a restore:

`m`

Normally, restored files retain the modification times that they had when they were archived. This option changes the modification times to the time of the restore. This is the opposite of its behavior with the `cpio` command.

> **Tip ✎**
>
> `tar`'s default treatment of modification times during a restore is the opposite of `cpio`'s.

`o`

This option tells `tar` to make you the owner of any files that you restore. This is the default behavior for users other than root. Unless this option is used, files extracted by root take on the user and group identifiers saved in the `tar` file.

`p`

By default, `tar` normally does not restore all file attributes. File permissions are determined by the current `umask` instead of the permissions of the original files. Also, the `setuid` and `sticky bits` are not restored for any files not owned by the user. This option tells `tar`

to use the permissions of the original files, including any special attributes such as `setuid`. (You must be root to set the `setuid` and `sticky bits` on other users' files.)

Some Other Neat Things About tar

`tar` has many options, and you should read the manpages to find them all. They can come in very handy.

Finding everything that's under the directory

Sometimes things underneath a directory are not what they seem. If you are creating "one last archive" of a directory before deleting it, you might want to follow any symbolic links that you come across. This is what the `-h` option is for. Make sure you've got lots of tape!

Using tar to move a directory

As discussed earlier, `cpio` has a built-in command to move directories. The problem is that many people do not remember its syntax when the time comes. However, you also can use `tar` to move a directory. You do this by first `cd`'ing to one level above the directory you are going to move:

```
$ cd old-dir ; cd ..
```

You then use `tar` and a set of parentheses to create a subshell that "untars" the directory into its new location. (Note the use of the `p` flag to ensure that `tar` creates the new directory with the same permissions as the old one.)

```
$ tar cf - old-dir | (cd new-dir ; cd .. ; tar xvpf - )
```

The `-` option for `tar cf` tells it to send its data to `stdout`. (We omit the `v` option to prevent writing the filenames to the display twice.) The `-` option on the `tar xvf` tells it to

look at `stdin` for its data. Surrounding the `cd old-dir ; tar xvf -` with parentheses creates a subshell so that the directory *old-dir* is extracted into *new-dir*.

> **Tip ✎**
>
> I have seen people try to move a user's home directory by `cd`'ing into that directory and creating a `tar` of "*". The problem with this is that it does not include the "." files such as *.profile*, *.cshrc*, or *.emacs*. I have then heard the person say, "Oh, I need to use `.*`, not *!". Remember always, and never forget, that the expression "`.*`" matches the string `. .` (the parent directory). *That means the archive also includes the directory above it.* That's why it is much easier to go a level above, and `tar` the directory. (Another way to do this would be to make an archive of "."). I prefer the former because it shows what directory the files came from.)

The syntax may seem a bit difficult, but it is very portable. It could be made a little shorter by saying:

```
$ cd parent ; tar cf - old-dir | (cd new-parent ; tar xvpf - )
```

Warning ☠

In this example, *parent* is the directory above the *old-dir*, and *new-parent* is the parent directory of the new location. For example, if you were moving */home1/fred* to */home2/fred*, *parent* would be */home1*, *old-dir* would be *fred*, and *new-parent* would be */home2*. Make sure you mean what you type. One of the problems with `tar` is that you get very familiar with typing `tar cvf`. Then one day you need to do a `tar xvf` and accidentally type a `c` instead of an `x`. Guess what happens. Your archive is ruined, and there is no way to fix it. This is one of the most common questions on Usenet, and there's never been a good answer for it.

Restoring to an alternate location

If you make your `tar` archives with relative pathnames, restoring to an alternate location is very easy. Simply change directories to something other than the original mount point (e.g., */home1*), and start the restore from there. `tar` creates directories as needed.

Tip ✎

If you did not create the `tar` archive with relative pathnames, you can use GNU `tar` to take off the leading slash.

Read the `cpio` section about relative pathnames and why they are important.

[7] Yet another reason why you should be using `gtar` if you are performing regular system backups with `tar`.

Backing Up and Restoring with the dd Utility

As far as backup utilities go, the `dd` utility is about as featureless as they come. However, it is uniquely suited for certain applications.

Basic dd Options

The basic syntax of `dd` is as follows:

```
# dd if=device of=device bs=blocksize
```

The preceding options are used almost every time you run `dd`; they are explained in the following sections.

Specifying the input file

The `if=` argument specifies the input file or the file from which `dd` is going to copy the data. This is the file or raw partition that you are going to back up (e.g., `dd if=/dev/dsk/c0t0d0s0` or `dd if=/home/file`). If you want `dd` to look at `stdin` for its data, you don't need this argument.

Specifying the output file

The `of=` argument specifies the output file or the file to which you are sending the data. This could be a file on disk or an optical platter, another raw partition, or a tape drive[8] (e.g., `dd of=/backup/file`, `dd of=/dev/rmt/0n`). If you are sending to `stdout`, you don't need this argument.

Specifying the block size

The `bs=` argument specifies the block size, or the amount of data that will be transferred in one I/O operation. This value is normally expressed in bytes, but in most versions of `dd`, it can also be specified in kilobytes by adding a `k` at the end of the number (e.g., 10 K). (A block size is different from a blocking factor, like `dump` and `tar` use, which is multiplied by a fixed value known as the minimum block size. A blocking factor of 20 with a minimum block size of 512 gives you an actual block size of 10,240, or 10 K.) It should be noted that when reading from or writing to a pipe, `dd` defaults to a block size of 1.

Changing block size does not affect how the data is physically written to a *disk device*, such as a file on disk or optical platter. Using a large block size just makes the data transfer more efficient. When writing to a *tape device*, however, each block becomes a *record*, and each record is separated by an interrecord gap. Once a tape is written with a certain block size, it must be read with that block size or a multiple of that block size. (For example, if a tape is written with a block size of 1,024, you must use the block size of 1,024 when reading it, or you may use 2,048 or 10,240, which are multiples of 1,024.) Again, this applies only to tape devices, not disk-like devices.

Specifying the input and output block sizes separately

When specifying block size with the option `bs=`, you are specifying both the incoming and outgoing block size. Sometimes you may need different block sizes on each. This is done with the `ibs=` and `obs=` options. For example, to read a tape with one block size and create a tape with another, you could issue a command such as this one:

```
# dd if=/dev/rmt/0 ibs=10k of=/dev/rmt/1 obs=64k
```

Specifying the number of records to read

The `count=n` option tells `dd` how many records (blocks) to read. You can use this to read the first few blocks of a file or tape to see what kind of data it is, for example (see the following section for more information). You can also use it to have `dd` tell you what block size a tape was written in.

Using dd to Copy a File or Raw Device

You can use `dd` as a backup command because it can copy the bits in a file or raw device to another location. You can even pipe the bit stream through `compress`, allowing you to store a compressed copy of the data. (`dump`, `tar`, and `cpio` do not have this capability, although GNU `tar` does.) The best example of using `dd` as a backup command is the hot-backup script for Oracle, *oraback.sh* (see Chapter 16 for more information about *oraback.sh*). Since Oracle can use both raw partitions and files for its database files, the script cannot predict which command to use. However, `dd` supports both of them!

Using dd to Convert Data

The `dd` command also can be used to convert data from one format to another in one pass.

Converting data to go into another command

Again, this is done by using different input and output block sizes (`ibs=`, `obs=`). If a command, such as `restore`, can read only certain block sizes, and you have a volume that was written in another block size, you can use `dd` to read the volume, and pipe the results of `dd` into `restore`.

Converting data that is in the wrong format

Although you may think of `dd` as a bit copier, it also can manipulate the format of the data, such as converting between different character sets, upper- and lowercase, and fixed- and variable-length records:

`conv=ascii`

: Converts EBCDIC to ASCII

`conv=ebcdic`

: Converts ASCII to EBCDIC

`conv=ibm`

: Converts ASCII to EBCDIC using the IBM conversion table

`conv=lcase`

: Maps US ASCII alphabetic characters to their lowercase counterparts

`conv=ucase`

: Maps US ASCII alphabetic characters to their uppercase counterparts

`conv=swab`

: Swaps every pair of bytes; can be used to read a volume written in different byte order

`conv=noerror`

: Does not stop processing on an error

`conv=sync`

: Pads every input block to input block size (`ibs`)

`conv=notrunc`

: Does not truncate the existing file on output

`conv=block`

Converts the input record to a fixed length specified by `cbs`

`conv=unblock`

Converts fixed-length records to variable length

`conv=..., ...`

Uses multiple conversion methods separated by commas

Using dd to Determine the Block Size of a Tape

This is kind of a neat trick. If you tell `dd` to read one block of data and then write it to disk, you can look at the size of that block to see what the block size of the tape is. Since you don't know the block size, start by using the largest block size that your operating system supports for that device, which is usually 128 K or 256 K, although it could be higher:

```
# dd if=device bs=128k of=/tmp/junk count=1
```

This tells `dd` to read data, using a block size of 128 K, until it gets to the first interrecord gap. If the block size is smaller than 128 K, it stops there. If it's bigger than 128 K, `dd` interprets it as an I/O error and complains. Just increase the block size value and try again. (Try 256 K this time.) This process creates a file called */tmp/junk*. The size of that file is the block size of the tape!

Using dd to Figure out the Backup Format

Here's another trick. Use the same command as in the preceding section to create the file */tmp/junk*, then issue the command:

```
# file /tmp/junk
```

This uses */etc/magic* to determine the file type. If it is `tar` or `cpio`, it usually comes back and tells you so. If it can't guess the file type, it just says "data," which isn't very helpful.

> **Tip ✎**
>
> Another interesting use of `dd` is to combine it with `ssh` or `rsh`. Be sure to read the section "Using ssh or rsh as a Conduit Between Systems" later in this chapter.

---

[8] Of course, a tape drive is another raw device as well.

Using rsync

Think of `rsync` as simply a copy command that can copy between systems. It's most like `rcp` in its syntax, but it's also like the Windows `copy` command to some degree. However, it has gone beyond a simple copy program by adding features such as the following:

Copies links, devices, owners, groups, and permissions

> This means that `rsync` can copy everything properly from the source to the destination, including special files and all of the appropriate permissions. It can copy both hard links and soft links as well.

Can use any transparent remote shell, including ssh or rsh

> `rsync`'s default authentication mechanism is now `ssh`, but this can be easily overridden by changing the `RSYNC_RSH` variable to `rsh`.

Can run as authenticated or anonymous daemon

> In addition to authenticating via `rsh` and `ssh`, `rsync` can also run as a daemon in either authenticated or anonymous mode. The former provides a more secure authentication mechanism, and the latter works really great for mirroring.

Has advanced exclude options

> `rsync` can exclude files in the same way GNU `tar` does, using exclude strings on the command line or by creating an exclude file and specifying it with the `exclude-from` option. In addition, `rsync` can be configured to skip the same files that CVS would ignore.

Sends only changed blocks of changed files

This is the biggest difference between `rsync` and `rcp`—and `rsync`'s greatest feature—and a lot of people don't realize it exists. When updating the destination, the source and destination split each changed file into blocks and run two CRC checks against each block. Only those blocks of data whose CRC checks don't match are transferred. This allows `rsync` to keep large files that change a lot in sync across much smaller pipes.

Sends several changed files as one large file

Since `rsync` performs a lot of single file and subfile activities, it can bunch them together into a single large transfer to reduce latency.

Can delete files

This is another big difference between `rcp` and `rsync`. `rsync` can delete files on the destination that are no longer present on the source.

Many people, including myself, have not really thought of `rsync` as a backup utility. One reason for this is that it is really a synchronization tool, not a backup tool. This means that, without some sort of intervention, a subsequent run of `rsync` overwrites the backup with a bad copy of the original, or deletes from the backup a file that was deleted on the original. That doesn't sound like a very good backup tool, does it?

However, it doesn't take a whole lot of work to put some history behind `rsync`. If you save previous versions before you overwrite them with newer versions or delete them, `rsync` can make an excellent backup tool. This book provides two examples of using `rsync` as a backup utility. Chapter 5 discusses BackupPC, and Chapter 7 describes near-continuous data protection using `rsync` and related utilities.

Basic rsync Syntax

Here are the basic ways to run `rsync`:

```
% rsync source [ source ...] destination
```

This command copies one or more source files or directories to a destination directory on the same machine:

```
% rsync source [source ...] username@hostname:destination
```

This command copies one or more source files or directories to a destination directory on a different machine, authenticating using `rsh`, or `ssh` if the `RSYNC_RSH` variable had been set to `ssh`:

```
% rsync source [ source ...] username@hostname::destination
```

Since the most common use for `rsync` for backup purposes is to transfer an entire directory tree from one machine to another, let's show that as an example. We want to transfer the directory */home* to */backup* on *backupserver*. We want to back up everything under */home* (recursive, or `-r`); we want to back up soft links (`-l`); we want their times (`-t`) preserved, and permissions (`-p`) including owner (`-o`) and group (`-g`) preserved; and we want any special files transferred as well (`-D`). This command could look like this:

```
% rsync –rlptgoD /home backupserver:/backup
```

Luckily for us, the `rsync` team realized that these options were very common for backup and archive purposes, so they created a single `-a` option that means the same as –`rlptgoD`. So the following simple command is the same as the previous one:

```
% rsync –a /home backupserver:/backup
```

Let's add verbosity (`-v`) and compression (`-z`) to the command:

```
% rsync –avz /home backupserver:/backup
```

To be truly synchronized, we need to add the delete flag to our command:

```
% rsync –avz --delete /home backupserver:/backup
```

Now, every time `rsync` runs, it copies everything from */home* to */backup/home* and deletes any files on */backup/home* that aren't present in */home*. All we've got to do is add some type of history collector on the other end, and we've got ourselves a backup system!

> **Tip ✎**
>
> Be sure to read Chapter 7 on open-source near-continuous data protection systems and Chapter 5 on `BackupPC` to learn more about how to use `rsync` in a backup setting.

A few twists

All of these commands copy */home* and its contents to the */backup* directory on *backupserver*. That means they create */backup/home*. If what you want to do is copy the contents of */home* to */backup* and not create a */home* subdirectory, just add a trailing slash to the source directory:

```
% rsync –avz /home/ backupserver:/backup
```

This command does the same as the following command, just with fewer keystrokes:

```
% rsync –avz /home backupserver:/backup/home
```

By default, `rsync` commands authenticate using `ssh`. You can authenticate using `rsh` instead by changing the `RSYNC_RSH` variable to `rsh`. In addition, you can also tell `rsync` to connect to an `rsync` daemon running on another

machine by putting two colons instead of one after the hostname:

```
% rsync –avz /home/ backupserver::/backup
```

If the `rsync` daemon you're connecting to requires a password, you can specify that password using the `RSYNC_PASSWORD` variable.

rsync on Windows

`rsync` is really a Unix-style binary, but it can be run on Windows if you use a Unix emulator such as *cygwin*. However, all the hard work has been done, and some members of the `rsync` team have actually created precompiled packaged binaries that come with the *cygwin1.dll* file and an *rsync.exe* file. Instructions on how to run `rsync` on Windows, including how to run it as a service/daemon, can be found from the main `rsync` web page at http://samba.org/rsync/nt.html.

rsync on Mac OS

Using `rsync` on Mac OS is quite simple. The only thing you have to add is the –`E` or – `extended-attributes` flag that tells Mac OS to transfer the additional attributes that Mac OS files have. Basically, this is the option that tells it to transfer the resource forks. (The only odd thing is that –`E` was an existing option on `rsync` that meant to transfer the executable bit in a file that was being transferred.)

Restoring with rsync

Restoring with `rsync` is exactly the same as backing up with `rsync`, except you change the order of the command. Specify as the source the location that is normally the

destination, and specify as the destination the location that's normally the source, and you've got yourself a restore. Let's take the system from our earlier example, and reverse the source and destination directories:

```
% rsync –avz backupserver:/backup/home/ /home
```

This tells `rsync` to restore everything from */backup/home* on *backupserver* to */home* on the local server. Of course, you can specify a single file as well:

```
% rsync –avz backupserver:/backup/home/curtis/resume.doc /home/curtis
```

The real challenge with `rsync` restores is not the syntax of the command, it's keeping track of what files should be brought back and which files are actually the same corrupted copies that you don't want to restore. That is the responsibility of the backup program that you're using. If you were using a snapshot-like utility like the one covered in the book, you'd simply add something like `daily.1` to the string to get yesterday's version:

```
% rsync –avz backupserver:/backup/daily.1/home/curtis/resume.doc /home/curtis
```

You can read more about using `rsync` to make snapshots in Chapter 7.

Backing Up and Restoring with the ditto Utility

`ditto` is a Mac OS X recursive copying utility, which can also create archive files (like `tar` or `cpio`). What makes it interesting is that it's the one native tool with the ability to create full backups on all versions of Mac OS X since support for HFS+ features such as resource forks was added when the tool was brought forward from NEXTSTEP. (See the section "How Mac OS Filesystems Are Different" earlier in this chapter for more on HFS+.)

`ditto` can copy files and directories to one of three types of destinations: a directory, a ZIP archive file, or a `cpio` archive file. It does not support copying directly to tape. On the other hand, it doesn't come with Yet Another Archive Format, so you won't get stuck with backup archives in some format that might not be easily readable in a few years.

Syntax of ditto When Backing Up

The most common use of `ditto` is to make recursive copies of files and directories, like so:

```
$ ditto –V --rsrc src... dest_dir
```

The `–V` flag shows everything that `ditto` is copying. The `–-rsrc` flag ensures that HFS+ attributes and resource forks are copied (which is the default from Mac OS X 10.4 onwards). Extra HFS+ information is stored in AppleDouble format, where the data for a file named *filename* is kept in *._filename*.

Using `ditto` like this is a lot like using `cp –R`, with one big difference. Let's say you want to make a copy of a directory.

Using `cp –R` *`src_dir dest_dir`*, you'd end up with the contents of *src_dir* under *dest_dir/src_dir/*. With `ditto` *`src_dir dest_dir`*, the contents of *src_dir* end up directly under *dest_dir/*, which can be somewhat confusing if you don't expect it. Also, `ditto` creates *dest_dir/* if it doesn't already exist.

In most cases, `ditto` makes an exact duplicate of the source. However, there are a few things that `ditto` won't copy, in which case you'll be missing some information:

- Named sockets; see the `socket(2)` and `bind(2)` manpages (which don't appear to exist in Mac OS X 10.4 for some reason, although they do in earlier versions). However, sockets should be created dynamically by programs that use them.
- Named pipes (or FIFOs); see the `mkfifo` manpage. Fortunately, Mac OS X itself doesn't employ any named pipes.
- BSD flags; see the `chflags` manpage. Again, Mac OS X doesn't come with BSD flags set on any files.
- Extended ACLs; see the `chmod` and `fsaclctl` manpages. By default, filesystems don't have extended ACL functionality enabled.

Tip ✎

These are apparently limitations of the underlying bill-of-materials (or BOM) framework employed by `ditto`. (See the `bom`, `mkbom`, and `lsbom` manpages.) `mkbom` also doesn't get named sockets or pipes, and the BOM file format doesn't include fields for BSD flags or extended ACLs.

In addition to making straight copies of files and directories, `ditto` can copy them into an archive file. To create a `cpio` file (with optional `gzip` compression), use the command:

```
$ ditto –V --rsrc –c -z src_dir dest.cpgz
```

To create a ZIP file, use the command:

```
$ ditto –V --rsrc –c -k src_dir dest.zip
```

When creating a ZIP file, using the `--sequesterRsrc` flag stores extra HFS+ data in a directory named *__MACOSX*; PKZIP-compatible utilities (other than `ditto` itself) may handle this better than AppleDouble.

As when making recursive copies, *src_dir* is lost from pathnames stored in an archive file. To retain *src_dir* in the archived pathnames, use the `--keepParent` flag.

One thing you can't do with `ditto` is selectively archive only part of a directory's contents—for example, you can't use a filename pattern or make incremental backups. `ditto` is suitable only for archiving entire directory trees.

You can use `ssh` and `dd` to make backups to remote systems, the same way you can with `tar` or `cpio`. For example:

```
$ ditto –V --rsrc –c -k src_dir - | ( ssh remote_host dd of=dest.zip )
```

Tip ✎

Note that in this example, `ditto` can archive to standard output; it can also accept standard input as the source. It's possible this functionality could be used for tape-based backup and restore (if suitable tape device drivers are available), but this hasn't been tested.

The Options to the ditto Command

`ditto` is a very simple command, with relatively few options and a straightforward argument syntax. Here are some of the options you can use; refer to the manpage for more:

`-v`

Prints the name of each source directory as it's copied.

`-V`

Prints a line for each file and directory copied by `ditto`.

`-c`

Instead of copying the contents of the source directory to another directory, copies to an archive file. This is a `cpio` archive by default, unless the –`k` flag is used.

`-z`

Uses `gzip` to compress the `cpio` archive.

`-k`

Creates a compressed ZIP archive instead of a `cpio` archive.

`-X`

Prevents `ditto` from crossing partition boundaries when copying.

`--keepParent`

Includes the source directory in the pathnames saved to the archive.

`--rsrc`

Copies HFS+ attributes and resource forks, in addition to standard Unix attributes and data forks. This is the

default for Mac OS X 10.4 and later. Can also be specified as `–rsrcFork`.

`--norsrc`

Prevents `ditto` from copying HFS+ attributes and resource forks. This is the default for Mac OS X 10.3 and earlier, or for Mac OS X 10.4 and later if the `DITTONORSRC` environment variable is set.

`--sequesterRsrc`

Saves HFS+ data and resource forks in a directory named *__MACOSX*, instead of in AppleDouble format.

`--arch`

When making a copy of an application with support for multiple CPU architectures (what used to be called fat binaries, and which Apple now calls Universal applications), copy only the elements for the specified architecture. The architecture can be either `ppc` (for PowerPC) or `i386` (for Intel, a reference to the first Intel CPU supported by NEXTSTEP).

`--bom`

Copy only the items listed in the specified bill-of-materials file. (You can create a BOM file with `mkbom directory`; see the manpage for more.) BOMs are used in Apple's Installer packages and record permissions, ownership, and a checksum for each item installed by a package.

Syntax of ditto when Restoring

Restoring the contents of a `ditto`-created archive is done with the `–x` flag (for "extract"). To restore from a compressed `cpio` archive, use the command:

```
$ ditto –V --rsrc –x src.cpgz dest_dir
```

The destination directory is created if it doesn’t already exist. Note that the –z flag isn’t required; `ditto` automatically handles compressed `cpio` files.

To restore from a ZIP archive, use the command:

```
$ ditto –V --rsrc –x –k src.zip dest_dir
```

There’s nothing special about archive files created by `ditto`; you could run extractions from any `cpio` or ZIP file using `ditto`.

If you want to restore only selected parts of an archive, use either `cpio` or `unzip` directly because you have no way of specifying that with `ditto`.

Listing the files in a ditto archive

To list the files in a compressed `cpio` archive, use the command:

```
$ cpio –itvz < src.cpgz
```

To list the files in a ZIP archive, use the command:

```
$ unzip –lv src.zip
```

Comparing tar, cpio, and dump

A few years ago, John Pezzano from Hewlett-Packard did a paper comparing native backup products. It is the best one that I have seen, so I asked his permission to update it a bit to reflect changes in the utilities and include it in this book. Table 3-3 compares `tar`, `cpio`, and `dump`.

Table 3-3. Conversion of native utilities

Feature	tar	cpio	dump
Simplicity of invocation	Very simple(`tar c` *`files`*)	Needs `find` to specify filenames	Simple—few options
Recovery from I/O errors	None—write your own utility	`resync` option on HP-UX causes some data loss	Automatically skips over bad section
Back up special files	Later revisions	Yes	Yes
Multivolume backup	Later revisions	Yes	Yes
Back up across network	Using `rsh`/`ssh` only	Using `rsh`/`ssh` only	Yes
Append files to backup	Yes (`tar -r`)	No	No
Multiple independent backups on single tape	Yes	Yes	Yes
Ease of listing files on the volume	Difficult—must search entire backup (`tar -t`)	Difficult—must search entire backup (`cpio -it`)	Simple—index at front (`restore -t`)
Ease and speed of finding a particular file	Difficult—no wildcards; must search entire volume	Moderate—wildcards; must search entire volume	Interactive—very easy with commands like `cd`, `ls`
Incremental backup	Can use `–newer` or `find` if using GNU `tar`	Must use `find` to locate new/modified files	Incremental of whole filesystem only, multiple levels

Feature	tar	cpio	dump
List files as they are being backed up	`tar cvf 2>` *`logfile`*	`cpio -v 2>` *`logfile`*	Only after backup with `restore -t >` *`logfile`* (`dump` can show % complete, though)
Back up based on other criteria	Yes, with GNU `tar`	`find` can use multiple criteria	No
Restore absolute pathnames to relative location	Yes, with GNU `tar`	With `cpio –I`, or with GNU `cpio`	Always relative to current working directory
Interactive decision on restore	Yes or no possible with *`tar -w`*	Can specify new path or name on each file	Specify individual files in interactive mode
Compatibility	Multiple platform	Multiple platform with ASCII header, not always portable	Readable between some platforms, but cannot be relied on
Primary usefulness	System backup if GNU `tar`, otherwise individual user backup, transfer files between filesystems	System backup, transfer files between filesystems	System backup
Volume efficiency	Medium, usually limited to 10 K block size	Medium—usually only 5 K block size, but can specify larger size on some OSes	High—can usually specify up to maximum block size of device
Wildcards on restore	No	Yes	Only in interactive mode
Simplicity of selecting files for backup from numerous directories	Low—must specify each independent directory, subdirectories included	Medium—`find` options	None—backs up one and only one filesystem
Specifying directory on restore gets files in that directory	Yes	No—must use *path*/*	Yes

Feature	tar	cpio	dump
Stop reading tape after a restored file is found	No	No	Stops reading tape as soon as last file is found
Track deleted files	No	No	If you restore with `-r`, files deleted before last incremental dump are deleted
Filesystem efficiency	Better	Worst (files get a `stat` from both `find` and `cpio`)	Best
Likelihood that file exists in TOC but not in archive	Low	Low	Medium (because TOC is made first)

Standard backup utilities may not be very sexy or even full of features, but if you get to know them, they will always be there. Some of the “semi-native” commands (for example, GNU `tar`, GNU `cpio`) are also very helpful, but they are not always available. Therefore, a good working knowledge of the truly native commands can come in very handy when you’re in a jam or when someone hands you an unknown volume and says “Can you read this?”

Using ssh or rsh as a Conduit Between Systems

This section explains how to use `ssh` or `rsh` as a conduit between systems, especially when combined with the functionality of `dd` and some of the other commands that can read or write to `stdin`. Even if your backup tool supports remote devices, such as `rdump`, it usually does so using `rsh` authentication. If you understand this section, you could use `ssh` instead, bringing more security to your backups.

Most other backup commands can only read or write from `stdin`, whereas `dd` can do both at the same time. This makes `dd` very versatile and the only native backup utility that can be used to pass a stream of data from one command to another or from one system to a device on another system, using `rsh` or `ssh`. This can work either way.

If you want to read a backup on a remote device, the `restore`, GNU `tar`, and GNU `cpio` commands can read the remote device by simply giving it `remote_host:remote_device` as the device name. However, the native versions of `tar` and `cpio` do not support such an option. To do this, you simply `rsh` or `ssh` a `dd` command to the remote system and read its data stream on the local system.

```
# rshremote_host "dd if=device ibs=blocksize"| tar xvBf -
```

Remember that when reading a tape volume using `dd`, you normally have to specify a block size. If you do not, it uses a block size of 512, which generates an I/O error unless the tape volume was written with that block size. Also notice the quotes around the remote `dd` command. In this command, the quotes are actually not necessary, because the pipe is executed on the local system. In other, more

complicated commands, such as one where there is a pipe to be executed on the remote system, placing quotes around the remote command makes things work properly. (In this instance, they merely makes it more readable.)

Writing a backup to a remote device is a bit trickier. You may have to create a subshell[9] with embedded `rsh` and `dd` commands and pipe the output of the local backup command to that:

```
# tar cvf - . \ |(rsh remote_system dd of=device obs=block_size)
```

Putting parentheses around the remote command creates the subshell. Notice that you must specify the remote block size, and you need to be careful when doing so. If you want to create a volume that can be read by `tar`, make sure you use a block size that `tar` can understand, such as 10,240. (This is usually the biggest block size `tar` can read or write, and this is done by specifying a blocking factor of 20 in `tar`.)

If you are not able to use `rsh`, you may look into using `ssh` as a drop-in replacement for `rsh`. The `ssh` command uses a much more secure authentication mechanism and allows you to use the same type of commands `rsh` does without the security holes that `rsh` opens. However, using the remote device feature of GNU `tar`, GNU `cpio`, or `dump` assumes the use of `rsh`. If you are not allowed to use `rsh` but can use `ssh`, you can use commands like the following to integrate `dump`, `tar`, and `cpio` with `ssh`.

To read tapes on remote hosts:

```
# ssh remote_host "dd if=device bs=blocksize"| tar xvBf -
# ssh remote_host "dd if=device bs=blocksize" \
 | restore rvf -
# sshremote_host "dd if=device bs=blocksize"| cpio -itv
```

To create backup tapes on remote hosts:

```
# dump 0bdsf 64 100000 100000 - \
  | ssh remote_host "dd if=device bs=64k"
# tar cvf - | ssh remote_host "dd if=device bs=10k"
#cpio -oacvB | ssh remote_host "dd if=device bs=5k"
```

Some commands work with `ssh` if you just change the `rsh` environment variable to `/usr/bin/ssh`.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

---

[9] Your mileage will vary. Not all versions of Unix require you to create a subshell.

Chapter 4. Amanda

The purpose of this chapter is to give a brief technical overview of Amanda. We want you to understand how Amanda works, how it is different from other backup software, and how it can help you solve your data protection requirements. On the other hand, we don't want to overwhelm you with technical details that could be very specific to a particular setup or backup policy. Throughout this chapter, we provide links to the web sites where you can find up-to-date and easy to follow instructions and details about everything you need to know about deploying Amanda in production.

> **Tip ✎**
>
> This chapter was contributed by Dmitri Joukovski and Stefan G. Weichinger.[1] Dmitri has been solving backup and recovery challenges since the early '90s and he lives in Silicon Valley, California with his beautiful wife and children. Stefan loves to work in a collaborative environment and continues his quest to find one.

Amanda, the Advanced Maryland Automated Network Disk Archiver, is the most well-known open-source backup software. Amanda was initially developed at the University of Maryland in 1991 with the goal of protecting files on a large number of client workstations with a single backup server. James da Silva was one of its original developers.

The Amanda project was registered on SourceForge.net in 1999. Jean-Louis Martineau of the University of Montreal has been the gatekeeper and leader of Amanda development in recent years. Over the years more than 250 developers have contributed to the Amanda codebase, and thousands of users provided testing and feedback, resulting in a stable and robust package. Amanda is included with every major Linux distribution. As of April 2006, more than 20,000 sites worldwide use Amanda.

CS Department Indeed

When I worked at a university, the student email/shell machine was run by the computer science (CS) department. It had purchased a tape drive and tapes, and they had set up `tar` to back up the system. The system was hacked, and it went down hard. I worked for the campus IT department in a part-time network admin job, so I called the people in the CS department and asked them about their backups. They told me the most current backup was on the tape on the machine. I looked at the date. It was two years old.

We tried to restore from the tape any data we could get from */home*, but it was not backed up. It was a complete loss of data. The IT department took control/ownership of the server at that point. We had backups going hours after we had the students back online, and we tested the restores. The only good part of the loss of the system was that we were able to finally get rid of the old accounts that should have been deleted.

Scott Boss

Originally, Amanda was used in production mostly by universities, technical labs, and research departments. Today with wide adoption of Linux in IT at large, Amanda is found in many other places, especially where the focus is on applications deployed on a LAMP[2] stack. Over the years, Amanda has received multiple awards from users. For example, in 2005 it received the *Linux Journal* Readers' Choice Award for favorite backup system.

Amanda allows you to set up a single master backup server to back up multiple Linux, Unix, Mac OS X, and Windows hosts to a very large selection of tape, disk, and optical devices including tape libraries, autochangers, optical jukeboxes, RAID arrays, NAS devices, and many others. Figure 4-1 shows a typical Amanda network.

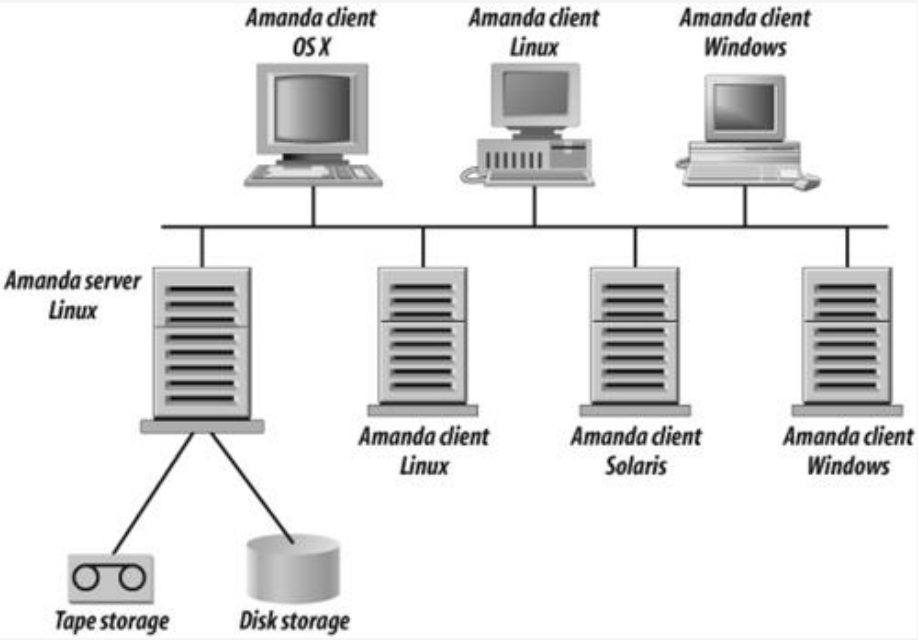


Figure 4-1. Typical Amanda network

Here are a few real-life examples of Amanda in production. One company uses three Amanda servers on CentOS in three countries to protect more than 30 clients on Solaris, Linux, and Windows. Different versions of Amanda have been in production for 9 years as of this writing. The total amount of protected data is more than 500 GB and data grows at 8 GB per week on average. One of the sites performs backup to disk only, and the other two back up to both disk and LTO autoloaders. System administrators recover files at least once per week because of users erasing files by accident. A few times over the years, the company lost servers because of failed hard drives, and Amanda came to the rescue for bare-metal recovery.

A major university in the United Kingdom has two Amanda servers on Fedora Core with more than 100 Linux (Fedora Core, Red Hat Enterprise Linux), Mac OS X, and Solaris clients with more that 2 TB of data. One of the Amanda servers is dedicated to backup of SAP and Oracle on Solaris.

A cinematographic post-production company has three Debian Amanda servers at two sites protecting 84 Linux and IRIX clients with 26 TB of data. It recovers files about twice per week due to user error. In three years of production, it had three instances of

total volume loss despite using RAID arrays, and Amanda was able to recover all three lost volumes.

Throughout this chapter, we use examples of real-life Amanda implementations. Based on feedback from many Amanda users with a variety of configurations and different levels of Amanda expertise, we believe that the key reasons for wide adoption of Amanda are:

Amanda simplifies your life as a system administrator because you can easily set up a single server to back up multiple networked clients to a tape, disk, or optical storage system.

Amanda is optimized for backup to disk and tape. Additionally, it enables you to write backups to tape and disk simultaneously. The very same data can be available online for quick restores from disk and off-site for disaster recovery and long-term retention.

Since Amanda does not use proprietary device drivers, any device supported by an operating system works well with Amanda. The system administrator does not have to worry about breaking support for a device when upgrading Amanda.

Amanda uses standard utilities such as `dump` and GNU `tar`. Since these are not proprietary formats, data can be recovered with readily available standard tools—even without Amanda.

Amanda's unique scheduler optimizes backup levels for different clients in such a way that total backup time is about the same for every backup run. Amanda frees the system administrators from having to guess the rate of data change in their environments.

The Amanda project has attracted a large and active community that grows every day.

The total cost of ownership (TCO) for a backup solution based on Amanda is significantly lower than the TCO of any solution that uses proprietary backup software.

Amanda software has a source-code tarball and RPMs for most common versions of Linux, and is available from http://www.zmanda.com. Additionally, source code is available from SourceForge.net at http://sourceforge.net/projects/amanda.

Some older (but stable) versions of Amanda are packaged with all common Linux distributions, including Fedora Core, Red Hat Enterprise Server, Debian, Ubuntu, OpenSUSE, and SUSE Linux Enterprise Server, including releases for Itanium, IBM p-Series and even IBM S/390 and z-Series mainframes.

Amanda documentation including a quick-start guide and FAQ, written by users for users, is available on the Amanda wiki at http://wiki.zmanda.com.

Open Source to the Rescue

In 1999, I began consulting for a small service organization within a U.S. government department. They used about 40 Windows PCs and 3 Sun servers, the latter running Oracle. For backups they used two separate commercial products and were unhappy with each. A fourth Sun server was already purchased, and the tasks were being shifted around, including the Unix backups.

I was asked for suggestions for a replacement for their backup software before an additional copy was purchased and support contracts renewed. I did a bit of research and discovered Amanda. I installed it on my home systems, ran it for a week, and suggested it to my management. But as was common in that time, management would not consider free software. Who would they get support from? What if something went wrong and it was discovered that free software was being used for such an important function as backup? How good can it be if it is free? Thanks, but no thanks. We'll make the safe choice: pay thousands of dollars for software we are not happy with, just because it is sold by a large company.

So they migrated their backups to a different server with some difficulty. Meanwhile, without telling my management, I started a parallel backup system with Amanda using the oldest Sun server and a spare DAT drive. About a month later, the crisis happened. A directory tree from several weeks earlier was needed. I was not involved in the recovery but I thought it was a good chance to compare recovery times from the two systems. Within about 20 minutes, I had used Amanda recovery

to get what I thought they were seeking and, copied it to a directory on their system under */var/tmp*.

From the other camp I heard much cursing and hair pulling all morning. In the afternoon I ended their torture and, pointing to the */var/tmp* directory, asked, “Is this what you need?”

Later I learned that the problem with the commercial backups was that the backup tapes were keyed to the backup server. Restores could only be made from the same server. The data they needed had been made on the previous backup server that now had neither installed software nor license. The backup tapes were basically worthless.

Management then decided to give Amanda a try as their primary backup system. Eventually they also backed up the PCs using Amanda. The last time I checked, Amanda was still in use in that department.

Jon LaBadie

Summary of Important Features

Let’s start with a brief overview of Amanda’s architecture. This will help you understand the most important concepts in Amanda functionality.

Client/Server Architecture Using Nonproprietary Tools

Amanda is designed to handle large numbers of clients and data, yet is reasonably simple to install and maintain. As a matter of fact, it takes more time to order a pizza than to configure an Amanda server with two Linux clients and one Windows client and to start a test backup. A white paper available at http://amanda.zmanda.com/quick-backup-setup.html provides detailed information about configuring Amanda backup in less than 15 minutes.

Amanda scales well up and down, so small configurations—even a single client—are possible. Many users back up just a single

client that is also the Amanda server. On the other hand, many Amanda users back up hundreds and even thousands of filesystems (there could be multiple filesystems per protected system) to a large tape library with multiple drives.

The Amanda code is written in C (with some Perl and shell scripts), and the code is portable to any flavor of Linux and Unix including Mac OS X. Windows clients can be backed up today via Samba or via a Cygwin client, which is a Linux-like environment for Windows. The Amanda community is actively working on providing a native client for Windows. The new Windows client will take advantage of Microsoft technologies such as Volume Shadow Copy Service (VSS) that provide snapshots of a system's volumes, including snapshots of open files.

The biggest advantage of Amanda over any other backup software is that Amanda does not use any proprietary data formats. It uses standard operating system utilities such as `dump` and `tar`, or open-source utilities available in many operating systems such as GNU `tar`, `smbtar`, and Schily `tar`, and uses the same archive format on the media. Depending on which one is the best match for your filesystems, directories, and files, you can mix and match these utilities as you wish. Since you use standard utilities, you can be confident that they will always be available to you. Another advantage of using standard utilities is that in case of disaster recovery or any other emergency, you can recover your data even without Amanda. (We explain how to recover data without Amanda when we discuss Amanda restores.)

Because Amanda uses standard utilities, it provides the following:

- Backup of sparse files
- Backup of hard links
- No changing of file timestamp during backup
- Exclusions of files and directories

From the system-administrator perspective, it is very important that Amanda does not use any proprietary device drivers. Any

device supported by an operating system works well with Amanda. In practical terms, this means that Amanda supports a wide range of tape storage devices, and new devices are usually not difficult to add. Many tape changers, stackers, jukeboxes, and tape libraries are supported by using special tape changer scripts to provide truly hands-off and lights-out backup. Basically, if you can read and write to your tape drive and move tapes in your tape library with standard operating system commands such as `mt`, Amanda will work with your tape library. Because Amanda doesn't use proprietary device drivers, another benefit is that you don't have to worry about breaking support for a device when upgrading to the latest version of Amanda.

To understand Amanda architecture and inner workings, let's take a look at a simplified Amanda configuration and review an example of a backup cycle, illustrated in Figure 4-2.

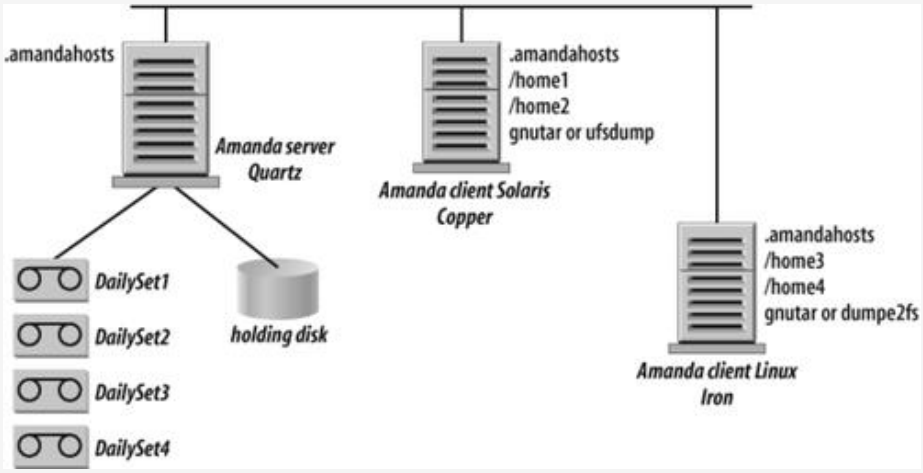


Figure 4-2. Amanda server with two backup clients

To simplify our discussion, let's assume that we have only two Amanda clients that run on two workstations: workstation *Copper* running Solaris and workstation *Iron* running Linux. Each workstation has two filesystems with users' data that we want to protect. Amanda server *Quartz* is installed on a different Linux host and, for simplicity's sake, we don't back up the Amanda server itself. (In your production and evaluation environments, you should always back up the Amanda server.) Let's also assume that we want to run a full backup once every four days and incremental backups between full backups.

Amanda is designed as a traditional client/server architecture. The Amanda server, also historically known as the *tape host*, is connected either directly or over the Storage Area Network to a tape drive or tape changer. Each client backup program is instructed to write to standard output, which Amanda collects and transmits to the tape server. The client/server architecture provides these benefits:

- It ensures scalability of Amanda from environments with a single client and CD-ROM to environments with hundreds of clients and large tape libraries with multiple tape drives and hundreds of tapes.
- It allows all configurations to be done on the Amanda server. Once the initial configuration of Amanda is done, you can easily add clients without worrying about breaking your tested backup procedures.
- It allows some CPU-intensive operations such as compression or encryption to be done on a client before sending backup images to the Amanda server. However, in some situations, for example when Amanda clients are running within virtual machines, these CPU-intensive operations can also be done on the Amanda server.

Considering the ever-increasing importance of security for backup data from a privacy and compliance perspective, let’s go over a brief overview of Amanda security.

Amanda Security

Amanda clients communicate with the Amanda server via its own network protocols on top of TCP and UDP. Amanda’s client/server communications do not suffer from the security holes inherent in the traditional `rmt` approach used by `dump`, such as using an *.rhosts* file in root’s home directory.

As in every other client/server setup, you should ensure that only your own and trusted Amanda server is able to communicate with Amanda clients. Amanda achieves that by using the file

.amandahosts. You can see that in Figure 4-2, there are three *.amandahosts* files, one on the Amanda server *Quartz* and one for each Amanda client. On the client side, you have to add the name of the Amanda server (or Amanda servers if you prefer the same host to be protected by multiple Amanda servers) and the Amanda user that is allowed to back up the client. For example, the . *amandahosts* file for Linux client *Iron* in Figure 4-2 should have the following entry:

```
quartz.zmanda.com     amandabackup
```

That tells the Amanda client *Iron* to let Amanda server *Quartz* communicate with user amandabackup.

During restores, you need access to an Amanda server. For the configuration presented in Figure 4-2, the *.amandahosts* file on the tape server *Quartz* should have the following entries:

```
iron.zmanda.com       root
copper.zmanda.com     root
```

These entries tell the Amanda server to allow the root user on each client to run restores. For security reasons, Amanda was designed to allow only the root user to restore data.

For stronger data transport security, Amanda can also use OpenSSH. This allows Amanda to protect the transfer of data between clients and backup servers with strong authentication and authorization mechanisms. Amanda also features an abstracted secure communication API that enables developers to easily add different communication plug-ins between backup server and client. Even with a single backup server, Amanda can use different communication mechanisms for different clients.

To protect data on the backup media itself, Amanda provides the ability to encrypt backup data with symmetric or asymmetric encryption algorithms (using either `aespipe` or `gpg`). Encryption is very expensive in terms of CPU utilization, which is why the Amanda encryption can be done either on the server or the client. (Do it wherever you have more CPU cycles available.) In addition to relieving the Amanda server CPU, client site encryption also ensures security of data on a wire, which could be important for backing up remote clients. Because of CPU

constraints, you might choose to encrypt only certain data. Amanda is flexible enough to configure data encryption for a single directory or even for a single file. If `aespipe` and `gpg` don't match your encryption requirements, Amanda will work with your custom encryption utilities.

> **Warning ☠**
>
> Amanda does not manage encryption keys. A system administrator should take care to safeguard the keys and make them available during restore.

Amanda works with Security-Enhanced Linux (SELinux), and it also works reasonably well with common types of firewalls between Amanda servers and clients as long as you select UDP and TCP port ranges during the initial setup. Please check installation and configuration details for firewall setup at http://wiki.zmanda.com.

To conclude this brief overview of Amanda security, we want to emphasize that the flexibility of the security configurations allows Amanda to fit well into the security policies and processes of most IT environments, including organizations with strict security requirements.

Holding Disk

You might recall that Amanda is actually an acronym, and D in Amanda stands for disk. To explain how Amanda moves data from the client to its final destination on tape or disk, you need to know the holding disk.

Figure 4-2 shows that the Amanda server *Quartz* has a holding disk attached. A *holding disk* is one or several directories on any filesystem that is accessible from the Amanda server. It could be as small as a single 10 GB directory on the Amanda server drive or as large as 5 to 10 TB on a fibre-attached RAID array. As the name suggests, the holding disk is used as a cache to store backup data from all Amanda clients. Each set of backup data

from a client filesystem or a client directory is just a bunch of files on the holding disk. Later, an independent Amanda process flushes individual backup images from the holding disk to tape or virtual tape at the maximum throughput possible to keep the tape drive streaming. Using a holding disk as a staging area for backups has several benefits.

Modern tape drives are very fast. Even gigabit networks cannot feed backup data from a single client through the Amanda server to modern tape drives fast enough to avoid *shoe-shining*, which reduces throughput and shortens the life of both the media and the drive (see Chapter 9 for details on shoe-shining). The holding disk collects data from all clients and as soon as the first backup is complete, it starts feeding data to tape as fast as the Amanda server can push it. However, many users prefer to complete backups of all clients before they start flushing data to tape.

A holding disk can accept data streams from multiple clients in parallel to overcome the sequential nature of a tape. Instead of writing one backup to tape after another, you can configure multiple backups running in parallel and make full use of your available network bandwidth, thus reducing total backup time. If the network becomes your bottleneck for performance, you can reduce total backup time by adding another NIC to the backup server or dedicating a separate network for backups. The use of multiple holding disks can also improve overall backup performance.

Using a holding disk provides additional safety in case you have a bad or wrong tape—or no available tape at all. Your backup is complete even if you forget to insert a new tape before taking the day off. It also provides a backup when a media error occurs during a backup run or the backup media runs out of space.

Amanda supports different algorithms to move the data from the holding disk to the media. Of course, your chosen algorithm will impact the effective use of the tape.

Amanda supports multiple holding disks so that backup images from different clients can be sent to different holding disks. This increases the scalability of Amanda and provides better load

balancing for I/O because holding disks can be on different controllers.

New Amanda users often ask how large the holding disk should be. In a typical “full and incrementals” backup cycle, most backups are small incrementals, so even a modest amount of holding disk space can provide better flow of backup images to a tape. A good rule of thumb is that there should be enough holding disk space for the two largest backup images at the same time, so that one image can be coming into the holding disk while the other is being written to tape. For example, if in Figure 4-2 the full backup of both filesystems for *Copper* is 50 GB and the full backup of both filesystems for *Iron* is 30 GB, the optimal capacity of holding disk on *Quartz* should be at least 80 GB. If that is not practical, any amount that holds at least a few of the smaller incremental backups is better than no holding disk at all. With today’s low disk prices, a good-sized holding disk is well worth the investment.

On the other hand, some Amanda users have significantly larger capacities for their holding disks. For example, a very large Japanese manufacturing company has four Amanda servers running on Solaris and BSD protecting more than a hundred Amanda clients on BSD, Windows, Linux, HP-UX, and Solaris running Oracle. One of its holding disks is on a RAID array with total capacity of 4 TB. Fast arrays and Amanda servers with high I/O allow streaming throughput from holding disk to tapes at approximately 120 Mb per second.

The flexibility of Amanda allows configurations without a holding disk, but then backups can be written to tape only sequentially instead of in parallel to the holding disk. Obviously, the lack of holding disk significantly reduces backup performance.

If the holding disk is for temporary storage of backup files, how does Amanda decide what to send to the holding disk in the first place? Let’s take a look at Amanda’s unique way of scheduling backups.

Backup Scheduling

Most backup products provide basically the same backup scheduling. The system administrator configures software to perform a full backup on Sunday, every other Sunday, or the last day of the month, with different levels of incrementals between full backups. The biggest problem with this approach is that it does not provide any load balancing. You have to make sure that enough resources are available to manage peak demand for backup server CPU, network, and I/O during full backups. Since you perform full backups only once in a while, your resources are underutilized most of the time. More often than anybody wants to admit, the system administrator finds out on Monday morning that Sunday's full backup did not complete because there were not enough tapes available in a library. Other Mondays you might find that your full backups are still running, and users are calling you to kill all backups. Of course, you can figure out yourself how to achieve load balancing by instructing your backup software to distribute full backups among your clients throughout the week or month, but then you have to make sure that there are no changes in your environment; new clients break down your balancing schema.

Amanda's unique approach to scheduling optimizes load balancing of backups and simplifies your life. Instead of giving Amanda the exact instruction "Do a full backup every Sunday for clients A, B, and C, full backups on Wednesday for clients D, E, and F, and incrementals all other times," you just set up a few ground rules that control Amanda scheduling. For example, you might give Amanda the rule "Do at least one full backup within a 7-day period, and do incrementals all other days with a maximum time between full backups of 7 days." The maximum time between full backups is called the *dump cycle*.

For any dump cycle you specify, Amanda finds an optimal combination of full and incremental backups from all clients to make the total amount of backup data per backup run as small as possible and consistent from one backup run to another. To find such a balance, Amanda uses the following considerations:

- The total amount of data to be backed up as reported by each client based on the amount of data changed since last

backup

- The maximum time between full backups (dump cycle) you specified
- The size of backup media (tape or disk) available for each backup run

To calculate the optimal backup level, Amanda starts every backup run with an *estimate phase*. Every Amanda client runs a special process to determine which files have changed and the total size of all changed files. The estimate phase can take some time, especially with many clients and filesystems. If some filesystems are not very dynamic and files don't change much, you can tell Amanda that, thus saving time during the estimate phase. After collecting data from all clients, Amanda goes into the *planning phase* and calculates the optimal combination of full and incremental backups for all clients.

Figure 4-3 shows how Amanda schedules backups for the clients from Figure 4-2, assuming that each home directory is 100 GB, the data change rate is 15 percent, and the dump cycle is 4 days. For simplicity, let's assume that Amanda writes each backup run to a new tape labeled DailySet1 through DailySet4 and that all incrementals are level 1 (level 0 is usually defined as a full backup), meaning everything that changed since the last full backup.

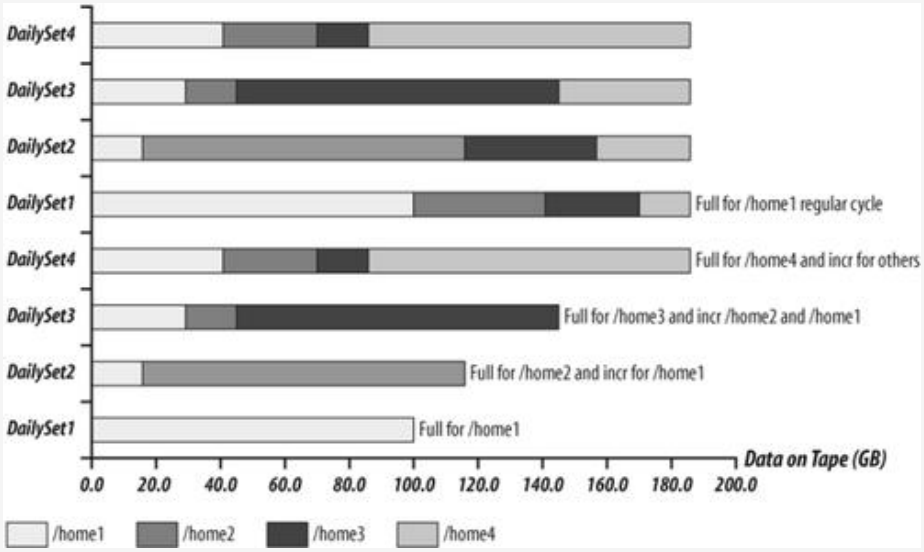


Figure 4-3. An example of Amanda scheduling

For each run, Amanda schedules a full backup for the total amount of data divided by the number of days in the dump cycle. Since the dump cycle is 4 days, for DailySet1, Amanda does the full backup for $^1/_4$ of the data, in this case */home1*. For DailySet2, Amanda does a full backup for another $^1/_4$ of data, in this case */home2*, and an incremental backup for */home1* which is 15 GB (15 percent of 100 GB). For DailySet3, Amanda does a full backup of */home3* and incrementals for */home1* and */ home2*. After the initial startup period of four days, Amanda runs a full backup for one of the */home* directories and incremental backups for all the others. Let's calculate the total amount of data on each DailySet tape (see Table 4-1).

Table 4-1. Example backup sizes (GB)

Directory	DailySet1	DailySet2	DailySet3	DailySet4	DailySet1	DailySet2	DailySet3	DailySet4
/home1	100.0	15.0	27.75	40.84	100.0	15.0	27.75	40.84
/home2		100.0	15.0	27.75	40.84	100.0	15.0	27.75
/home3			100.0	15.0	27.75	40.84	100.0	15.0
/home4				100.0	15.0	27.75	40.84	100.0
Total	100.0	115.0	142.75	183.59	183.59	183.59	183.59	183.59

It is trivial to calculate the total amount of data for DailySet1 and DailySet2. For the third backup run of */home1*, we have to consider that 15 percent of data was backed up on DailySet2,

which means 15 percent of 100 GB has changed again (see Figure 4-4).

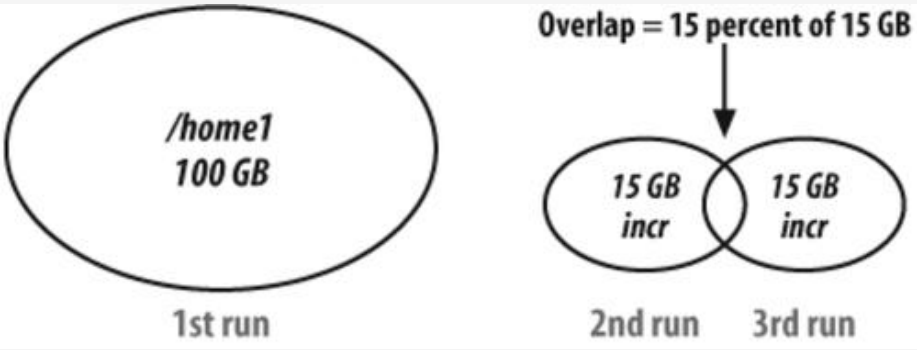


Figure 4-4. Overlap in data in subsequent backup runs

To avoid double counting, we have to subtract the small overlap area from 30 GB. So for DailySet3, the size of the incremental for */home1* is 30 GB – (15 GB×15 percent) or 27.75 GB. Following the same logic, for DailySet4 the incremental for */home1* is not 45 GB. It's 45 GB – (27.75 GB×15 percent) or 40.84 GB. This example is admittedly a mathematical oversimplification. In reality, Amanda uses all nine levels of incremental backups to optimize the total amount of data on tape.

In addition to a traditional schema with full backups and incrementals in between, Amanda also supports:

- Periodic archival backup, such as taking full backups off-site
- Incremental-only backups with full backups done outside of Amanda, such as very active areas that must be taken offline or no full backups at all for areas that can easily be recovered from vendor media (such as the installation CD for an operating system)
- Always doing full backups of databases that change completely between each run or any other critical areas that are easier to deal with during an emergency if they are a single-restore operation

It's easy to support multiple configurations on the same Amanda server, such as doing traditional full backups and incrementals on a weekly basis and also doing additional monthly full backups for off-site storage. Multiple configurations can run

simultaneously on the same tape server if there are multiple tape drives.

When choosing the length of your dump cycle, remember that shorter dump cycles such as three to four days make restores easier because there are fewer incrementals, but they use more tape and require more time to back up. Longer dump cycles allow Amanda to spread the load better over multiple tapes but may require more steps during a restore. More information about how to choose a reasonably balanced dump cycle depending on amount of data, tape drive capacity, and so on is available at http://wiki.zmanda.com.

Now, let's take a look at Amanda tape management.

Tape Management

Each tape should be labeled before use with the command `amlabel`. There is a default template for labels, and you can define your own label templates. Labeling prevents overwriting of tapes with valid backup images and allows the Amanda server to keep track of all tapes that were labeled. Amanda starts a new tape for each backup run (for example, each nightly backup) and does not provide a mechanism to append a new run to the same tape as a previous run.

Based on your backup retention policy, Amanda keeps track of the expiration date for each labeled tape, and Amanda reuses that tape for new backups after it has expired. However, you can configure Amanda not to reuse specific tapes. You might choose to never expire some backup images and use Amanda for creating archives. (Amanda's support for optical media is very useful for archiving.)

For backups of large amounts of data, Amanda supports using multiple tapes in a single backup run. For example, backups from clients A, B, and C can be written on one tape, and backups from clients E, F, and G can be written to another tape.

In the past, Amanda could not span multiple tapes for a single backup image and system administrators had to break large

filesystems into smaller chunks, such as several directories. As of version 2.5, Amanda can span multiple tapes. The size of the backed-up images is no longer restricted to a single tape, and there is no need for the system administrator to artificially segment data into parts that can fit into a single tape.

Device Management

Amanda does not use any proprietary drivers for tape or optical devices. You have to make sure your tape devices are configured as nonrewinding devices (for example, */dev/nst0* and */dev/nst1*). You also have to select the *tapetype definition* specific to your tape-drive technology. Many default tape definitions are provided with Amanda. Here is an example of tapetype definition for LTO-3:

```
define tapetype LTO3-400-HWC {
     comment "LTO Ultrium 3 400/800, compression on"
     length 401408 mbytes
     filemark 0 kbytes
     speed 74343 kps
}
```

Note that Amanda does not use the length of the tape value. It tries to write to the tape until it gets an error.

You have to select a tape changer script for your tape changer. Examples of tape definitions for most commonly used tape drives and details about configuring tape drives and tape changer scripts are available at http://wiki.zmanda.com.

For a long time, Amanda has provided the ability to use disk as the target media for backup. Dedicated directories are used as virtual tapes called *vtapes*. You work with vtapes exactly the same way you work with real tapes. For example, you have to label vtapes before they can be used by Amanda. You might use vtapes and backup to disk as a way to test and evaluate Amanda before you decide to invest in an expensive tape library. Furthermore, backup to disk is now a viable option for production from a cost perspective. You get all the benefits of having a backup of your data without the challenges of managing finicky tape drives.

A most interesting scenario is the use of tapes and disk at the same time. Amanda provides a functionality called RAIT

(Redundant Array of Inexpensive Tapes). Initially RAIT was designed to increase redundancy. This is the same technology as RAID, where data is striped over several disks. Amanda supports RAIT with two-, three-, and five-tape sets.

A three-drive RAIT set writes two data streams and one parity stream and gives you twice the capacity, twice the throughput, and the square of the failure rate (for example, a 1/100 failure rate becomes 1/10,000 because you lose data only if two tapes are faulty or not available). Similarly, a five-drive RAIT set gives you four times the capacity and four times the throughput. A two-drive RAIT set duplicates the output stream, and each output stream can have either the same or different media targets. If you have the same media targets (for example, two tape drives), you get exact copies of your backup data, called *clones*. You can keep one clone on-site for occasional restores and take another clone off-site for disaster recovery.

If you have different media targets, you can keep your backup data on disk for two to three weeks for occasional restores. For long-term retention, you have a copy on tape. Most restores happen within 10 days after a file has been lost, and the ability to restore data quickly from disk becomes very important.

---

[1] Because so many technical writers before us wrote excellent articles about Amanda, we want to give due credit to John R. Jackson, Alexandre Oliva, →leen Frisch, Paul Bijnens, and many others who contributed to the wealth of published knowledge about Amanda. Many of their ideas made it into this chapter.

[2] The acronym LAMP refers to a set of open-source software tools commonly used together to run dynamic web sites or servers. LAMP stands for Linux, Apache, MySQL, Perl, PHP, Python.

Configuring Amanda

Now let's take a look at how to configure Amanda backup. Detailed instructions about how to install and configure the Amanda client and server are available from http://wiki.zmanda.com. Here we provide a configuration roadmap. The preferred way to install Amanda is from RPMs found at http://www.zmanda.com, but if you want to compile from source, this section presents the basic procedures for installing the client and the server.

> **Tip ✎**
>
> While one machine can be both a client and a server, there is no need to perform both procedures; installing the server normally installs the client.

To compile the Amanda client from source:

1. Create an *amandabackup* user in the disk group. This user must be the same as the backup user.
2. Unpack, compile, and install Amanda from the source archive. Specify configure options for a client-only compile, and install.
3. Add Amanda-related entries to */etc/services* and the */etc/xinetd.d* directory, and restart `xinetd`.
4. Specify which servers are allowed to connect by editing the *.amandahosts* file to enable authentication between client and server.

To install the Amanda server, you can also use RPMs. If you want to compile the server from source:

1. Create an *amandabackup* user in the disk group. The backup user should belong to the user group that has read and write access to the media.
2. Unpack, compile, and install from the source archive. Specify options for a client-and-server install.
3. Add Amanda-related entries to the */etc/xinetd.d* directory, and restart `xinetd`.

Once the server is installed, configure the Amanda server:

1. Create the Amanda configuration directory */etc/amanda/<config name>* (in case you use Zmanda packages).
2. Copy a sample of *amanda.conf* into */etc/amanda/<config name>*.
3. Create a *disklist* file in */etc/amanda/<config name>*.
4. Edit the *.amandahosts* file to enable authentication between client and server.
5. Edit configuration files *amanda.conf* and *disklist*.
6. Add client configuration to the *disklist* file.
7. Set up the device (create device nodes or directories) if you are using virtual tapes.
8. Label the media (tapes or vtapes) using `amlabel`.
9. Configure a `cron` job to schedule Amanda backup runs.
10. Run `amcheck` to verify that there are no problems with configuration, client/server communications, the holding disk, or the tape.

Figure 4-5 shows the configuration files for our sample network.

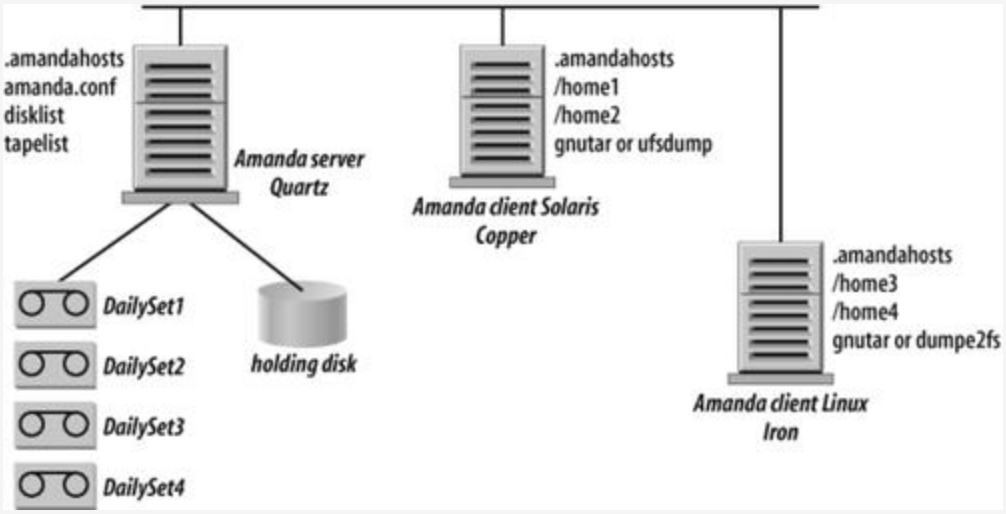


Figure 4-5. Amanda configuration files

The most important file for configuring Amanda setup is *amanda.conf*. The example file is quite large (more than 700 lines, which is why we don't include an example here; see http://wiki.zmanda.com for details), but fairly self-explanatory. This file defines how you do your backups, using settings that include:

- Local settings, such as name of your organization, email address to send backup completion report and warnings, and location of Amanda directories
- Device and network settings, such as names of your tape devices, tape type definitions, tape changer script, tape labeling template, and network bandwidth for Amanda to use
- Backup policy settings such as dump cycle, specifics about incremental backups, number of backup runs per dump cycle, and number of tapes in rotation
- Holding disk locations with capacities available to Amanda
- Instructions for how to perform backups, such as whether to use `dump` or GNU `tar` and details about

indexing data, encryption, and compression

The *disklist* file specifies what to back up. For example, to back up the */home* directories for clients *Iron* and *Copper* in Figure 4-5 requires the following disk list entries (DLEs):

```
# hostname diskname dumptype
Copper /home1 stable
Copper /home2 stable
Iron   /home3 normal
Iron   /home4 normal
```

Dumptypes are defined in the *amanda.conf* file. They specify backup-related parameters, such as whether to compress the backups, whether to record backup results in */etc/dumpdates*, the disk's relative priority, and exclude lists. Here are sample definitions for the dumptypes `normal` and `stable` that we used for *Copper* and *Iron* entries in the *disklist* file:

```
define dumptype  normal {
comment "gnutar backup"
holdingdisk yes # (on by default)
index yes
program "GNUTAR"
priority medium
    }

define dumptype  stable {
comment "ufsdump backup"
holdingdisk yes # (on by default)
index yes
program "DUMP"
priority medium
    }
```

Many parameters in *amanda.conf* have default values, but the wide variety of parameters available for editing gives you full control over your backup environment.

A new Amanda user should plan on a learning curve of about two to four weeks before having a full production backup. It does not mean that a novice user will spend the

whole month studying the Amanda wiki and reading the source code. As a matter of fact, it takes less than 15 minutes to configure an Amanda server with two Linux clients and one Windows client and to start a test backup. A white paper available at http://amanda.zmanda.com/quick-backup-setup.html provides detailed information about the "Start Amanda backup in 15 minutes" benchmark. However, you should plan to allocate some time to get comfortable with Amanda functionality and to test your restores several times before going into production. For large sites, it is a good idea to add one or two clients every day until all clients are protected by Amanda.

So far, we have described the most typical situation: an Amanda client configured on the system to be protected. However, a system administrator may decide to mount a filesystem via NFS or Samba on the Amanda server and have the Amanda client running on the server back up these networked filesystems.

We All Make Mistakes

I had just started using Amanda to back up the main server at a small web shop. I was feeling overconfident and wanted to listen to CDs on the server, so I tried to connect the audio cable of the CD player to the sound card of the machine—without shutting down. I got the CD connected but happened to bump the SCSI cable between the three RAID5 disks and the RAID controller. This was the backup server, so when the RAID card refused to play, and the server wouldn't boot, I was looking at a bare-metal recovery of the backup server. Doh!

Fortunately, the folks who built the server, VALinux, were able to put me in touch with an engineer of theirs who knew the voodoo to tell the RAID card to ignore the fact

that the disks were in an unknown state and to just bring them back online.

I've since had similar situations with a different RAID card that refused to play due to power loss during boot. That time I had to drive a very long way just to get the floppy with the RAID software and docs on it.

Moral 1: Yes, even you can make mistakes.

Moral 2: Keep your RAID docs and media with your RAID card.

Jason in California

Backing Up Clients via NFS or Samba

Compared to the traditional approach of using an Amanda client on the system to be protected, there are several advantages of backing up via the Network File System (NFS) or Samba:[3]

- You don’t need to install and configure Amanda client software on the system to be protected. More systems can be added to the list of protected systems by modifying files on one system.
- A native Amanda client may not be available for a particular operating system. An NFS/SMB approach gives you the ability to back up such an operating system.
- You use significantly less CPU and memory resources on the protected system.
- If the important data from the protected systems is already available via NFS or CIFS, this approach provides better integration with the company IT policy.

However, consider the trade-offs of this approach:

- Think about the security issues of the mounting mechanism, whether SMB or NFS. You will not be able to use the mechanism provided by Amanda to encrypt the data in transit from the client to the server. You will also need to understand the security implications of running NFS or Samba on the system you want to protect in the first place. For example, if you do not want to run an NFS server all of the time on the system you want to back up, you will need to craft a synchronization scheme in which the NFS server on the system starts running just before the backup starts.

- Access privileges need to be carefully worked out. The Amanda server needs both read and write privileges on the NFS mount point. Read permission is necessary during the backup phase. Write permission is necessary during a restore.
- You will not be able to use local filesystem-based mechanisms for optimizing the backup process. For example, you will not be able to use `dump` or `xfsdump` on a filesystem being accessed over the network.
- You cannot back up any open files being accessed via Samba. You cannot back up extended file attributes via Samba.

Backing Up Using NFS

To back up using NFS, you need to install and configure an NFS server on the target system, and an NFS client on the Amanda server. At this point, export the filesystems to be backed up (by listing them in the */etc/exports* file of the client system). Make sure that the Amanda server can access all the files that need to be backed up. In many cases, this means turning on the `no_root_squash` option on the NFS share being backed up so that the Amanda server can access all files. Note that the hostname in the corresponding disk list entry should be the system where the NFS share is being mounted (not the client system). For example, in the sample network in Figure 4-6, the hostname in the disk list entry would be *Quartz*.

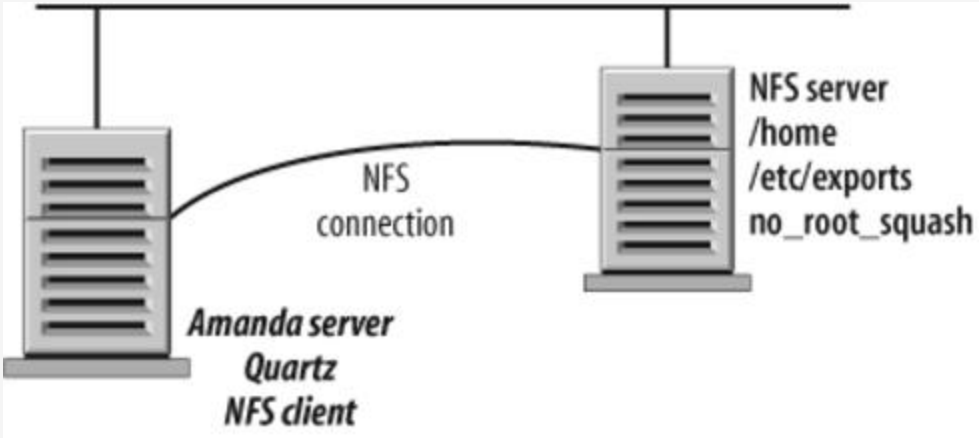


Figure 4-6. Configuring NFS-based backup

Backing Up via Samba

To back up using Samba, install the Samba client on the Amanda server. You don't have to explicitly mount the remote filesystem. Amanda is well integrated with the `smbclient` utility (an `ftp`-like client that is used to access SMB/CIFS resources on servers). It uses the `-T` option to create `tar`-compatible backups of all the files on an SMB/CIFS share. Amanda clears the archive bit of the files on the Windows-based target it backs up, enabling the incremental backup process.

A user must be created on the Windows system with full access rights (read/write) to the share. (In the example network in Figure 4-7, the user is *amandabackup*.) Amanda connects to the share as this user. If the user does not have full access, incremental backups will not work, and the whole share will be backed up every time (because the archive bits are never reset). Note that if any other program on the Windows system resets the archive bit of a file, Amanda will not back up that file during an incremental backup.

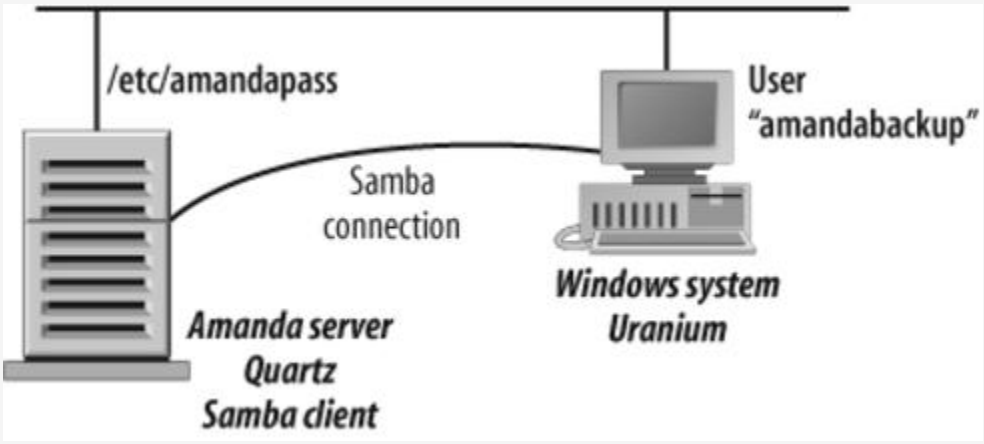


Figure 4-7. Backing up a Windows-based system using Samba

In addition to the standard Amanda configuration, you need to create the file */etc/amandapass* on the system running the `smbclient` utility. This file contains authentication information to access specific Windows shares. Also note that the hostname in the corresponding disk list entry refers to the system running `smbclient`, not the Windows system being backed up. In Figure 4-7, it would be Amanda server *Quartz*.

Many Amanda installations protect Windows servers and PCs in production. For example, the Radiology Department at a large Midwestern university has been using Amanda since 1999. In the past, their Amanda server ran on IRIX, AIX, and Solaris, but the current Amanda server runs on Linux with indices replicated to another server. They back up more than 70 Linux, Solaris, IRIX, Mac OS X, and Windows clients with around 4 TB of backup data. The holding disk is 1.4 TB, and the dump cycle is 90 days. All Windows clients are protected using Samba. Several times per month, they recover files because of user error or hard-drive failures and have never lost data because Amanda was always able to recover lost files. The next section describes Amanda recovery.

[3] Samba is an open-source implementation of the SMB protocol, which is also called Common Internet File System (CIFS). SMB/CIFS is commonly used on Windows systems.

Amanda Recovery

`amrecover` and `amrestore` restore Amanda backups. `amrecover` restores files using an interface that allows browsing of the backup file index to a certain date and selecting files to restore. Of course, to use `amrecover`, you should enable indexing of backup files when you specify the *dumptype* in *amanda.conf*. After you select files, Amanda finds the required tape, looks for the backup image, decompresses the image if required, brings the image over the network to the client, and pipes it into the appropriate restore program with the arguments needed to extract the requested files. In case you have to restore your files from incremental backups, Amanda specifies the correct order of the tapes. For security, `amrecover` must run as root on the client, and you should list root as the remote user in *.amandahosts* on the Amanda server.

Full filesystem recovery should be done with `amrestore`, which retrieves whole filesystem images from tape.

`amrecover` can be done on any client including the Amanda server. `amrestore` can be done only on the Amanda server. You have to use `amrestore` when you don’t have a backup index.

If your backup policy specifies backup of everything including the operating system, you can do bare-metal recoveries with Amanda. Here’s the procedure:

1. Replace the disk.
2. Boot with LiveCD (for example, Knoppix).
3. Format and partition the disk.
4. Use `amrestore` on the Amanda server to restore everything to a new disk, including the operating

system (you might need to restore incrementals as well).

5. Create a boot loader on the new disk.

The Amanda tape format is simple so that in case of emergency, you can restore data without any Amanda tools. The first tape file is a volume label with the tape volume serial number and date it was written. It is in plain text. Each file after that contains one image using 32 KB blocks. The first block is an Amanda header with the client, area, and options used to create the image. As with the volume label, the header is plain text. The image follows, starting at the next tape block, until end of file.

Because the image header is text, it may be viewed with these commands:

```
# mt rewind
# mt fsf NN
#dd if=$TAPE bs=32k count=1
```

In addition to describing the image, the Amanda tape format contains text showing the commands needed to do a restore. Here's a typical entry for the */home2* filesystem on *iron.zmanda.com*. It is a level 1 dump done without compression using Solaris `ufsdump`:

```
AMANDA: FILE 20060418 copper.zmanda.com /home2 lev 1
comp N program /usr/sbin/ufsdump
```

To restore, position the tape at start of file and run:

```
# dd if=$TAPE bs=32k skip=1 | /usr/sbin/ufsrestore -f... -
```

To retrieve an image with standard Unix utilities if `amrestore` is not available, position the tape to the image, then use `dd` to read it:

```
# mt rewind
# mt fsf NN
#dd if=$TAPE bs=32k skip=1 of=dump_image
```

The `skip=1` option tells `dd` to skip over the Amanda file header. Without the `of=` option, `dd` writes the image to standard output, which can be piped to a decompression program, if needed, and then to the client restore program.

If RAIT is used as the media, a shell script using the commands `dd` and `mt` must be used to restore data from the tapes without using Amanda commands. As with any backup system, you should test and retest your restore procedures so you know exactly what to do when disaster strikes.

Community and Support Options

We've described Amanda's core functionality, but there is more to learn. For in-depth information about Amanda monitoring, reporting, self-checking, encryption, and many other features, see the following resources.

Amanda is the only open-source backup software with enterprise support, available as a subscription from Zmanda, Inc. (http://www.zmanda.com). Zmanda also offers indemnification to select buyers of its Amanda Enterprise Edition subscription from any intellectual property infringement issues. In addition, professional services are available from Zmanda and several other organizations for installing and configuring Amanda.

Amanda documentation written by users for users is available at the Amanda wiki at http://wiki.zmanda.com. Ease of remote editing by multiple users, an ongoing archival of changes, and search capability are key features of this wiki. The Amanda community uses various collaboration tools including Amanda forums at http://forums.zmanda.com. Amanda users also have a very friendly mailing list at amanda-users@amanda.org with archives available at http://groups.yahoo.com/group/amanda-users.

Future Plans

One of the main challenges in IT today is the overall security of systems. Since security is such a fundamental part of backup (especially when people lose unencrypted tapes), the Amanda community plans to continue hardening all aspects of Amanda security.

There is a fundamental shift in the backup industry, with disk becoming the primary media for backups. Even though Amanda was designed for backup-to-disk from the very beginning, the Amanda team plans many backup-to-disk improvements, such as providing multiple simultaneous backups and restores from disk.

Many Amanda users have a constant battle with overwhelming data growth. Amanda has to be up to the task, and the Amanda team is working on increasing scalability and performance.

Wide adoption of open-source products (especially Linux) brings Amanda to production environments with Oracle, MySQL, SAP, and many other applications. Many users successfully deploy Amanda in such demanding environments, and the Amanda team is working on an application API that will simplify backup of those applications.

Amanda has always strived to simplify the life of the system administrator, and the project continues to work on the simplification of installation, administration, and recovery while giving the system administrator full control of how to do backups.

As you can see from this short list, the development of Amanda continues addressing of real-world requirements of real people. The Amanda development team and the

Amanda community will further maintain and enhance this powerful and well-known software suite.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 5. BackupPC

Most small businesses and home networks now have a mix of machines to back up: desktops, servers, and laptops, often with a variety of operating systems. To this challenge, add the fact that they do not have tape robots or the money for a high-end tape solution. Now throw in the problem of laptops, which by design are mobile and may not be on the local LAN in the middle of the night when backups typically run.

Most open-source solutions are not designed to solve this particular set of problems. Some don’t support disconnected or Windows DHCP clients. Others don’t allow users to schedule their own backups or restores. Finally, most of these solutions require installing additional software on the machine to be backed up, creating challenges with machines that are not centrally administered, such as Linux systems or Mac laptops.

Tip ✎

This chapter was contributed by Don Harper, Global Linux Engineer for JP Morgan/Chase. Don is a second-generation computer professional (his father was lead systems analyst for Shell Oil in the ’70s) and has been making his living working with Unix systems since 1987, from work in startups to large multinationals.

BackupPC Features

BackupPC is an entirely disk-based backup and recovery system. It offers a number of advantages, some of which are available only with BackupPC:

Support for any client OS

: By using standard tools that either come with the base distribution or can be easily added to the system, it is possible to support a wide range of clients. In addition, there is no need to install any client software beyond standard system utilities (`tar`, `ssh`, `rsync`). Adding new client operating systems becomes easy, especially if the client is a Unix derivation such as Mac OS X.

User control of and access to backups through web interface

: Every major OS has a web browser, so using a web interface is another way to speed the process of supporting new operating systems. The web interface should be designed to give as much control to the client as possible and do it securely. A user should be able to request a restore without having to find the backup operator, and easily browse and restore individual files. However, the user should not be able to see another user's machine.

Support for DHCP and disconnected clients

: Again, by using standard utilities, BackupPC supports DHCP clients as long as the client is registered with a name service such as DNS, Active Directory, or LDAP. The problem of hosts that roam from the network should be handled by testing for the client on the network and not raising an error until a set amount of time has passed.

How BackupPC Works

The BackupPC model has one user per client. This fits the usage pattern of the type of environment it was specifically designed for: backing up several users' PCs (hence the name). This should typically be the user who owns the data on the machine. In the case of a large file server, it should be an administrator. BackupPC emails the owner if it cannot back up the client after a configurable time, and the owner can control restores using the web interface. The following list describes how BackupPC works:

Direct to disk

> BackupPC stores all its backups directly on disk. Identical files across any directory or client are stored only once, which dramatically reduces the server storage requirements. These files are stored in a *disk pool*. In addition to the disk pool, the backups are in a directory tree organized by host, then by backup with hard links to the disk pool.
>
> BackupPC also has a nightly process to reclaim space from the disk pool that is no longer referenced by any backups, which helps keep the overall disk usage from growing out of bounds. This is an automatic process that the administrator does not have to configure.

Support for any client OS

> The server portion of BackupPC is designed to run on a Unix-style system using Perl and `mod_perl` under Apache for best performance, but it can be run on any web server that supports Perl and running Perl CGIs. (It does require either `mod_perl` or setuid Perl.) The server should have a large disk or RAID disk for backup storage.

As for clients, almost any Unix or Unix-like OS can be easily backed up. Most modern versions of the commercial Unix variants (Solaris, AIX, IRIX, HP-UX) have `tar`, `compress`, `gzip`, `rsync`, and `rsh` and/or `ssh` either in the base distribution available on the Web. Other Unix operating systems (Linux, FreeBSD, OpenBSD, NetBSD, Mac OS X) also have these tools.

Windows clients can be backed up in a few different ways. If the local policy prevents additional software being loaded, BackupPC can use part of the Samba suite (http://www.samba.org) to back up SMB shares on the client. If software can be installed locally, then `rsync` together with the `Cygwin` tool set (http://www.cygwin.*com*) can be used on the client.

Support for native tools

BackupPC uses standard Unix tools for its tasks. This includes programs such as `perl`, `tar`, `rsync`, `compress`, `gzip`, `bzip2`, `zip`, `apache`, and `samba`. This makes porting the server to a new OS much smoother than trying to port C code. BackupPC does not use a database or catalog to store backup information. Instead, it uses the disk tree to store this information. This means that upgrading the operating system of the BackupPC server (or upgrading the BackupPC application itself) is painless.

User control of backup/restores through web interface

The Web is the main interface for BackupPC. After the initial configuration, there is no need to have command-line access to the server to administer BackupPC. The web interface is written in Perl and has been designed to run either under `mod_perl` or normal CGIs running with setuid Perl. The interface allows users to log in and control on-demand backup and restores. The user can

request a one-time backup, a full backup, or an incremental backup. If the user needs to recover a file, there are a few options. Individual files can be downloaded simply by selecting them. Groups of files or directories can be restored back in place, or the user can download the files as a `tar` file or, if configured, as a ZIP file. The user has full control over which files or directories to restore and where to restore them. A history feature displays which files changed during each backup in each directory.

Support for DHCP and disconnected clients

Since BackupPC's clients are referenced by hostname, if the network being backed up uses DHCP and has dynamic name resolution enabled, nothing further needs to be done for the BackupPC server to back up DHCP clients. If this is not the case, and the clients are Windows machines, BackupPC can be configured to search an address pool for the clients, locating them via their `smb` hostname.

If the client is not online during its normal backup period, the BackupPC server does not generate an error unless a set period of time has elapsed since the last successful backup. At this point, the server emails the owner of the client and reminds him to ensure the machine is on the network for a backup. (The server can also email any errors to the administrator.)

Clients that live on a remote LAN can be backed up locally assuming there is network connectivity between the sites. This means that clients connected via Virtual Private Network (VPN) can be backed up. If the user does not want to back up at that point, a trip to the web GUI can cancel the current backup. Clients can also optionally block out times for no backups to permanently fix the issue. BackupPC uses `ping`'s round-trip time to

determine whether a client is on a remote network, and won't back up the machine if the round-trip time is longer than a configurable setting.

Backup pooling

If many clients use the same OS, many duplicated files will be backed up. Keeping multiple full backups increases the number of duplicate files, which increases the storage requirements for the server. BackupPC stores a directory tree per client backup but checks to see whether any file has been stored before from any client. If one has, BackupPC then uses a hard link to point to the existing file in the common disk pool, saving a great deal of space. In addition, BackupPC can optionally use compression to save more space. For example, on a server with nine clients, eight Linux machines, and one Windows 2000 machine, backing up only system configuration and user files, the server has 195 GB backup up before pooling and compression, but disk usage is actually below 40 GB. This is for two full backups and two weeks of daily backups per client. Pooling of common files and compression typically reduce the server's disk storage requirements by factors of six to eight.

Easy per-client configuration

After the administrator has defined what the site backup policies should be, it is very easy for her to override any configuration option on a per client basis. This allows great flexibility on what, when, and how to back up a client. There are no classes of clients per se, but this can be achieved by `symlinking` configurations for clients from a master for the "class."

Installation How-To

The BackupPC server runs on Unix or Linux under the Apache web server using `mod_perl`, which are not difficult requirements. The BackupPC server is not designed to run under Windows because Windows filesystems do not readily support Unix-style hard links.

Prerequisites dictate how the backups are performed. Table 5-1 lists the high-level requirements and suggestions for tools that can meet the requirements.

Table 5-1. BackupPC requirements

Function	Suggested tool	Needed for	Other tools	Notes
HTTP server	Apache	Control GUI	Any CGI-capable HTTP server	
CGI	`mod_perl`	Speed		Optional
Perl	Perl 5.8	Server written in Perl		
`tar`	GNU `tar`	Archive files in a `tar` container for transferring to server	Any command-line `tar` program	If `tar` method is used
`rsync`	GNU `rsync`	Transport from client to server	Any command-line `rsync` program	If `rsync` method is used
`smbclient`	Samba	Transport from client to server		If SMB method is used
`ssh`	OpenSSH	Transport layer		If `tar` or `rsync` methods are used

The amount of disk space needed depends on the amount and type of the data to be backed up. The more diverse the data, the more disk space is needed. A RAID subsystem is not required, but it would be a good idea for performance, data protection and scalability reasons.

The server doesn’t need anything more than a 1.5 GHz processor with at least 512 MB of RAM. Since much of the backup computation is compression, more memory and a faster processor are needed for large numbers of clients. Of course, faster disk technologies and clean networks also improve performance.

Security Versus Ease of Use

As is often the case, there is a trade-off between ease of use and security. For a home installation, ease of setup is usually a higher priority than security. The opposite may be true for a division of a global company.

The default configuration has the BackupPC service running as its own user on the server machine and using preshared `ssh` keys without passwords to connect to the clients as root, or SMB using a password in the case of Windows. This may not fit with local site security policies. Other options include setting up an `rsync` server on the client using stored passwords, or a two-step process of `ssh` to a low-privileged client account followed by `sudo` with a configuration that allows only `rsync` (or `tar`) to be executed.

All of the BackupPC processes on the server run as one user ID. This user ID should have limited privileges on the system. The install process ensures the chosen user ID has the correct permissions on the data storage areas. If a new user ID is used, setting up the web interface requires a few extra steps. If the existing ID of the web server is used, there are fewer setup steps, but the system will be less secure. If you use a new user ID, create it before starting the install process.

Basic Sizing

The amount of space needed is highly dependent on the number and type of clients being backed up. The more homogeneous the clients, the more effective the disk pooling is, and the less disk you need. The more diverse the clients, the less effective the disk pooling is, and more space is needed. Additionally, if the data is relatively static, less space is needed for the incrementals. With a lot of data change, the incrementals are larger.

The retention policy also affects storage requirements; more full backups and longer retention times naturally increase storage requirements.

What NIS+?

A client called me after he'd had some problems with NIS+. The manager there (engineer, not IT) wanted to clean up unwanted files on his */var* partition and found some handy "log" files to delete. (He removed the transaction logs for NIS+ on the NIS+ master.) NIS+ immediately recognized the fact that its transaction logs were no longer there and stopped.

When I got on-site, I asked the backup administrator for the backup tapes. They did weekly fulls and no incrementals. I found that all four weeks of tapes were incomplete; they'd been getting errors on backup for more than six weeks and hadn't done anything about it.

I had to recreate the databases from whatever information I could find and fix the problem with their backup script. I also changed the output of the backup script so that it yelled a little louder when it had errors—hopefully loud enough to not be completely ignored.

Michael Rice

To determine the amount of space needed for backups, add up the disk usage of each client, then multiply by the

number of full backups configured. The resulting number is the amount of space required for the full backups (prior to pooling and compression). Next, estimate what percentage of data will change for each incremental. Multiply this by the total amount of data, then by the number of incrementals configured. Add this to the number for the full backups. Compression and disk pooling reduce storage requirements by a factor of six to eight, so divide the total by this factor. To provide headroom as client storage grows and as more clients are added, you might want to start with two to three times this amount of storage.

For example, consider 100 laptops backing up user data that averages 4 GB per client, with each incremental averaging about 0.4 GB. Storing 3 weekly full backups takes around 1,200 GB, and 6 incremental backups takes another 240 GB for a total of 1,440 GB of raw data. Because of pooling and compression, approximately 180–240 GB of storage are needed. To support growth of the user data or additional clients, 500 GB or greater capacity should be sufficient for current and future needs.

Because BackupPC uses hard links to compactly store identical files, the entire data store must be on a single filesystem. Using a RAID array or LVM setup allows this filesystem to be expanded over time as needed.

Installing BackupPC

Once the server is identified, and the prerequisites are installed and verified to work, it is time to get down to business. Head over to http://backuppc.sourceforge.net/, and download the latest `tar` ball. Do not use the beta unless there is a specific need for it. If there is a patch file, download it as well.

Move the `tar` ball into a working directory and unpack it. Change the directory to the newly created directory. If you have a patch file to apply, do that now:

```
$ patch -p0 < [path to patchfile]
```

Something went wrong if you see an error message like `Hunk #1 FAILED at 58`, and you should verify that the patch-file version matches the distribution you downloaded. If you are still unsuccessful, search or contact the mailing list for help (see the section "The BackupPC Community" at the end of this chapter for details).

The next logical step is to read the *README* file for any details that may have changed for this release. In addition, there is up-to-date and complete documentation in the *doc/* subdirectory.

Read through the *BackupPC.html* file, and note the specific requirements listed in the installation section. Generally, a few Perl modules need to be installed depending on how the system is configured.

Now, run the following command as root:

```
# perl ./configure.pl
```

This process inspects the system and asks for some installation information:

Full path to existing *conf/config.pl*

> This is only used for upgrades. Press Enter for a new install.

Are these paths correct?

> Verify that the list of programs have been fully qualified. If a program is not found but you are not planning to use it, it is safe to ignore it. If all the needed programs are listed, press Y; if there is a critical program missing,

press Ctrl-C to stop the installation. Fix the problem and rerun the configuration script.

BackupPC will run on host

The script guesses the hostname. Correct as needed.

BackupPC should run as user

This is the user ID that all of the BackupPC processes will run as on the server. Either create a user with no special privileges or choose one with limited privileges.

Install directory (full path)

This is where the BackupPC program and library files will be stored.

Data directory

This is where the data store will be located. This should be on its own partition and preferably on its own disk or RAID array.

Compression level

This the amount of compression used to store the backups. There is a trade-off between the amount of compression and speed. This value should be from 0 to 9, with 0 being no compression and 9 being the highest amount of compression with the most CPU usage. The default, 3, is a good middle ground.

CGI bin directory

The full path to the web server’s CGI bin.

Apache image directory

This directory will hold the image files for the web GUI. It should be a directory that the web server can display.

URL for image directory (omit http://host; starts with “/”)

This is what the last part of the URL needs to be to display something placed in the image directory.

At this point, most of the questions should be answered. The script asks whether you want to continue before actually modifying the system. Answering y here allows the script to create the necessary directory structure and installs the scripts to run. As with any install script, watch the output for any errors. Several *init* scripts are updated with settings based on the configuration responses but are not automatically installed. These scripts are located in the *init.d* subdirectory where the configure script was run. Copy the correct one into the correct location for starting the service on boot.

Note that starting in version 3.0 of BackupPC, the *configure.pl* script complies with the Filesystem Hierarchy Standard (FHS). One change is that all the configuration files are by default stored below */etc* rather than below the data store. As described earlier, any of the default locations can be changed when running the configure script. If you are upgrading for the first time to version 3.0 or later, the *configure.pl* script continues to use the original locations for the data store, configuration files, and program executables.

Installation packages

An alternative to the manual installation procedure described here is to find and install a BackupPC package specific to your operating system. Packages exist for Debian, Ubuntu, Gentoo, and others. For example, BackupPC can be installed on Debian with:

```
# apt-get install backuppc
```

Starting BackupPC

The BackupPC server is started by using the *init.d* script created as part of the installation. This means it is automatically started after a reboot.

Using the CGI Interface

By default, the CGI interface should be accessible via the URL http://localhost/cgi-bin/BackupPC/BackupPC_Admin. Depending upon your Apache setup, you might need to create an *htaccess* file for user authentication.

Configuration Files

Starting in version 3.0, the host and configuration settings can be edited using the CGI interface. You can also configure the server by manually editing the configuration and host files. To do this, `change directory` to the data directory defined during installation. Then `change directory` into the *conf* directory. In version 3.0 and later, the default configuration directory is */etc/BackupPC/conf*. In the *conf* directory, there are two files that must be edited before BackupPC is usable. The first file is the *hosts* file. It contains all the hosts that the server will back up. The format of the file is:

```
host  dhcp user moreUsers
```

where *host* is the hostname of the client, *dhcp* is set to 0 if the machine can be found via normal name lookups, or 1 if the service needs to look in the DHCP pool, *user* is the name/email of the primary owner of that machine, and *moreUsers* is a comma-separated list of users who are able to access this host via the web GUI.

The file *config.pl* in this directory is the master config file. As the name implies, this is a Perl file, and all variables are set using Perl syntax, which allows arrays to be used for values. This file defines the number of backups running (`$Conf{MaxBackups}`), the number of full backups to hold (`$Conf{FullKeepCnt}`), and the length of time between full backups (`$Conf{FullPeriod}`). Read this file; it is well commented and includes many settings that you may want to change.

Trust but Verify

At an oil company where I worked, each desktop workstation that had a local data drive also had a local tape drive. It was made *very* clear to everyone who received one that it was their responsibility to back up their own data. We provided scripts, `cron` jobs, and training upon request on how to do the backups. We came around quarterly to clean their tape drives for them, but we could not do the backups for them. (There were 3 admins covering 300 offices; it just could not physically happen.) Network backups were also impossible at that time.

One day we got a call from a highly paid geophysicist saying that his 9 GB disk drive (big for that time) had died. We replaced the defective disk drive and asked for his backup tape to restore his data. “What backup?” was his response. We sent the disk drive out for “data recovery.” This took a couple of months and cost over $20,000. Most of the data was recovered—all the work of this geophysicist for the last six months (when the drive was first installed). The geophysicist almost got fired for “not securing company assets.”

Although we didn’t change the practice of geophysicists being in charge of their own backups, we did look a little

closer and a little more often at what they were doing.

Jack

Per-Client Configuration

It is very easy to override the default settings per client. Under the data directory, there is a *pc* directory with one directory per client. To have custom client settings, simply create a new *config.pl* in the appropriate directory with the settings needed. As of version 3.0, per-client configuration files are stored below */etc/BackupPC/pc*.

The per-client configuration files are used to specify different transfer settings, passwords, exclude files, and number of backups to keep. For example, the main configuration file might specify the `XferMethod` as `smb` for the Windows client machines while the per-client configuration files for the Linux machines override `XferMethod` to `tar` or `rsync`. Almost any setting can be overridden in the per-client configuration file. The exceptions are any settings dealing with the server itself, such as the wakeup schedule.

The BackupPC Community

How big is BackupPC's installed base? Given the nature of an open-source project, it is very hard to give exact figures, but here are some for an overview. From September 2001 until late February 2006, there were over 87,000 downloads of the core product from SourceForge. These downloads don't include installations via packages for standard Linux distributions such as Debian. In that time, there have been over four million visitors to the SourceForge project page. The project ranks in SourceForge's top 500 projects (out of over 110,000 projects). On Freshmeat, BackupPC has a user rating among the top 50 projects (out of over 40,000). Sites of all sizes are currently running BackupPC, from home users to small businesses, nongovernmental organizations (NGOs), schools, universities, corporate departments, and large companies. Some of the largest BackupPC installations include a large school district with 1,500 clients and a division of a large company with 4,000 clients, each of which involve multiple servers with several terabytes of storage.

The BackupPC community is very responsive and helpful. There a few ways to join and become active in the BackupPC community, and there are many places to turn if you run into problems.

The BackupPC web site, at http://backuppc.sourceforce.net/, provides many helpful tools for the community. The documentation covers the configuration in depth and is kept current with any changes in the code. The SourceForge site also has a FAQ.

If the answers cannot be found there, searching or posting to the mailing list normally solves any issue. The main

mailing list is BackupPC@lists.sourceforge.net. The main developers are very active on the list, and they take time to help out new folks. The list itself is not high volume, but the quality of the discussion is very useful. The list is very forgiving of new users and works hard to help them out.

The developers respond fairly promptly, and, if the issue is a bug, work with the bug reporter to get the issue solved. When posting to the mailing list, try to include as much information as possible, such as the OS of the server and the client, a copy of the *config* files, and the error sections from the logs.

The Future of BackupPC

Features are steadily being added to BackupPC. Recent additions include a full CGI-based configuration editor and improved internationalization support. Currently, the big development activity is around `BackupPCd`, a side project being worked on in tandem with the work being done on the main server. `BackupPCd` is a client for the BackupPC server that will handle all the issues involved in dealing with the client. It will also implement its own transport protocol with the server. This protocol is being based on the `rsync` protocol, ensuring a reliable and efficient transport. While BackupPC will continue to support the existing transport mechanisms such as SMB, `tar` over `ssh`, and `rsync`, using `BackupPCd` will allow ACLs and other file metadata to be backed up, avoiding the need to install `cygwin` on Windows machines for `rsync`, allowing a uniform backup protocol across different client operating systems, and providing better performance.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 6. Bacula

Bacula is an open-source backup product designed to scale from single machines to enterprise networks. It is capable of backing up to any combination of disk, tape, or optical media. The server currently runs on a Linux or Unix host with clients available for a variety of Unix and Linux, post-Win95 SE Windows, and Mac OS X systems. A Windows-based server has been under active development. Bacula 1.40 marks its first release; as is the case with the initial release of any software, it is prudent to test it extensively before relying on it for production use.

> **Tip ✎**
>
> This chapter was contributed by Adam Thornton. Adam has worked in system administration, analysis, and architecture since 1989. Besides backup systems, his other passions include retrocomputing, mixology, and packing and carting with his dogs.

Bacula was originally written by John Walker and Kern Sibbald in 2000. John left the project not very long after its inception, and Kern, now the primary Bacula maintainer, worked on it alone from mid-2000 until Bacula's initial public release in April 2002. Since then, other developers have contributed additional work, augmenting Kern's continued involvement.

Bacula is available under the terms of a modified version of the GNU Public License version 2. The additional restrictions added to the GPL are available in the file *LICENSE* in the top-level Bacula source directory.

The project’s homepage is at http://www.bacula.org, and its downloadable files and CVS repository are hosted by SourceForge.

Bacula Architecture

Bacula is structured as a number of cooperating components, all of which use TCP sockets to communicate over a network connection. The use of TCP/IP as the intercomponent transport is essential to Bacula’s design philosophy because it allows components to be deployed on multiple or separate machines (according to capability or access to specialized hardware) and provides a ubiquitous method for intercomponent commands. The TCP transport can be wrapped with a standard Transport Layer Security (TLS) encryption layer to protect data while in transmission. TLS is discussed in more detail in the section “Advanced Features” later in this chapter.

Bacula Components

Figure 6-1 shows the structure of Bacula and how its components are related.

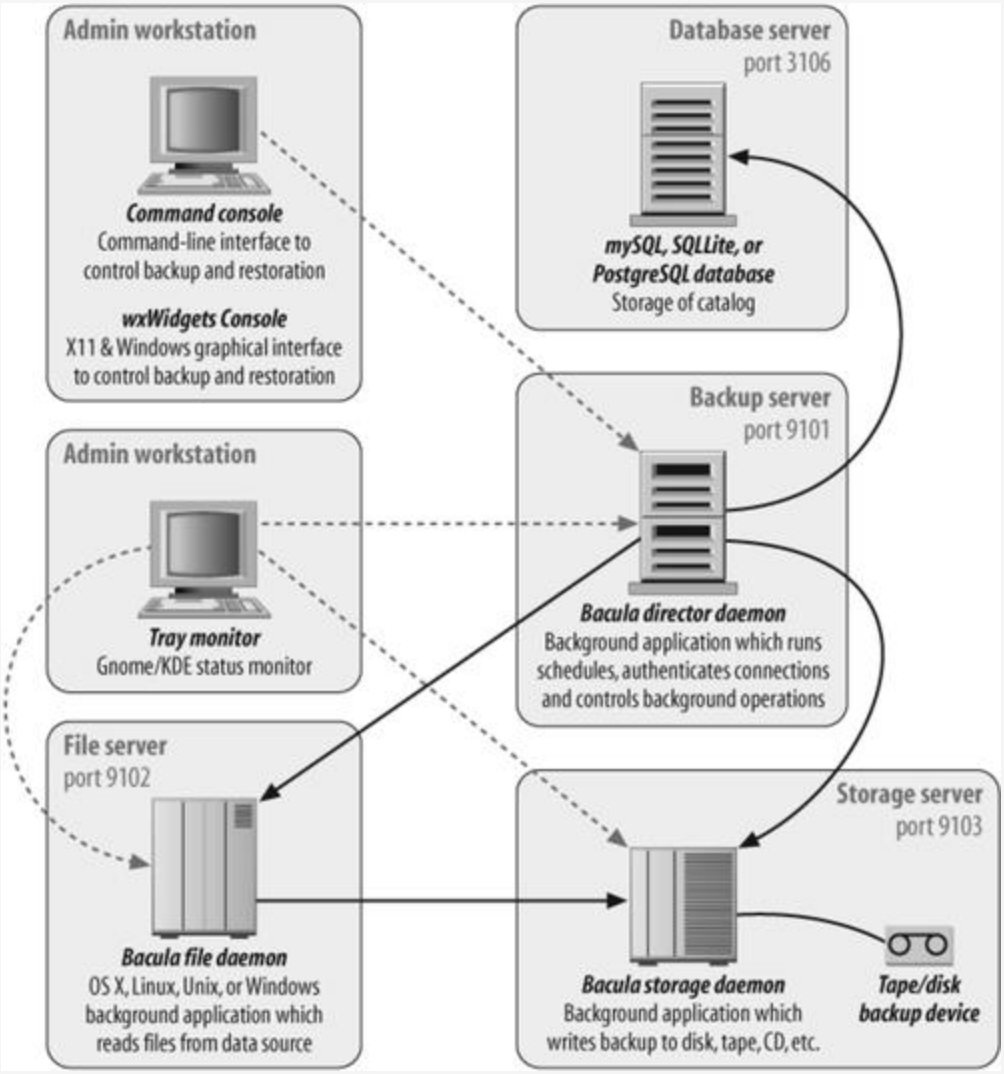


Figure 6-1. Bacula application interactions (Source: adapted from Bacula manual)

The separate application components illustrated here each provide a basic function within the Bacula environment. The following list identifies each component and describes the function provided to the overall application:

Director

The Bacula director is the application at the heart of a Bacula deployment: it manages media pool definition, scheduling, dependency tracking, access control, and reporting. It is responsible for nearly all policy-based

configuration. Most changes to Bacula configuration occur in the director's configuration file.

Database server

Another essential piece of the Bacula design philosophy is the *catalog*—an index of backed-up file storage locations—which is kept in a relational database and accessed by SQL queries and updates. Currently, Bacula supports three databases: SQLite, MySQL, and PostgreSQL.

Storage daemon

The Bacula storage daemon manages interaction with media used to store backup data and is the only part of Bacula that actually communicates with volumes (such as tape, disk, or DVD-ROM) used for backups. All other Bacula components use the storage daemon's TCP/IP interface to communicate with the storage daemon and are unaware of the actual methods used to store and retrieve data. The storage daemon manages mounting and unmounting of storage volumes, labeling tapes (using OCR barcodes if available), and management of automatic tape loaders/libraries (ATLs).

Administrative console

The administrative console provides the operator user interface for job management, message handling, and status information. Volume management operations (like manually labeling a new storage volume) are also handled from the console.

The console comes in multiple flavors. A line-mode console is available on all platforms, is the most well supported, and is the way most administrators choose to interact with Bacula. A GUI is available on systems with the wxWidgets libraries; the current GUI is a graphical wrapper over the line-mode interface, which makes

interactive file restoration somewhat easier and provides tab-completion and instant interactive help for commands. The line-mode console is often preferred, largely because it is easy to operate over low-bandwidth connections, making remote administration easier. The Bacula console does not have to run on the same machine as the director.

Tip ✎

One of the major current Bacula projects is to create a Python-based GUI, which will be much more extensible and flexible than the current wxWidgets GUI.

File daemon

The file daemon actually transfers the client's data to the storage server. It must be installed on each machine that is to be backed up. This piece communicates with the director to determine what client data is to be backed up and which storage daemon is to be used, and then transmits the data directly to the selected storage daemon, including metadata about the files it is sending (such as file size, access control parameters, and ownership information) as well as file contents.

Bacula also provides small status-indicator application-tray monitors for Windows, GNOME, and KDE (with some support for other windowing environments that support the free-desktop system-tray standard). The tray monitor is simply an icon that resides in the system tray and indicates whether Bacula components on this machine are idle, currently running a job, or in an error state. These status monitor icons can be expanded to give complete status information.

Interaction Between Components

Figure 6-2 shows the information flows between Bacula components in a typical backup job.

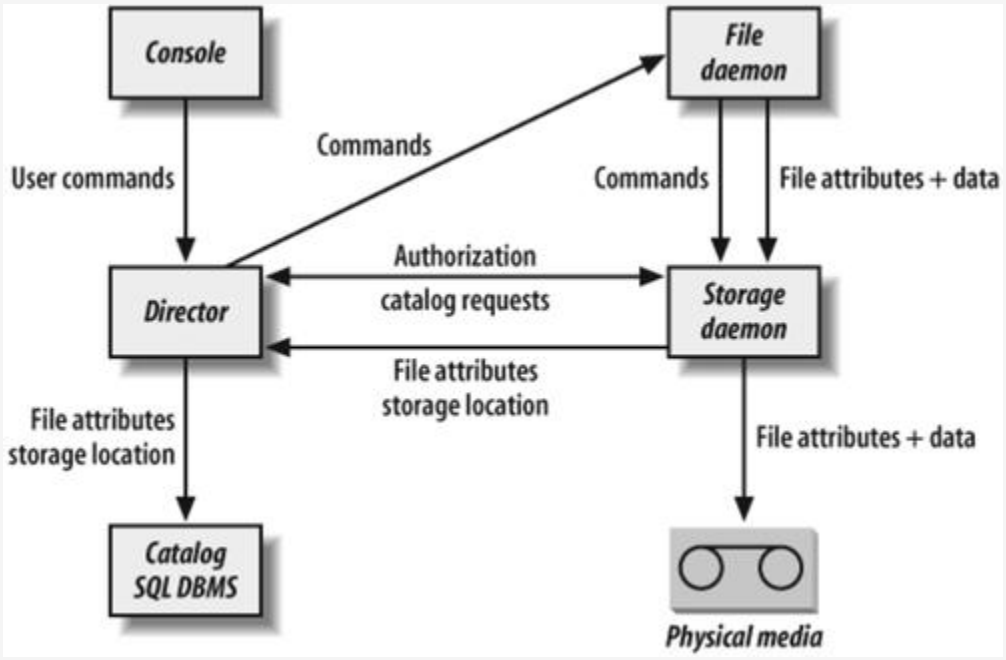


Figure 6-2. Information flow in Bacula

During execution, the director instructs the file daemon to communicate directly with the appropriate storage daemon, thus removing the director from the high-traffic path of actual backup data. The director maintains its own scheduling service so that jobs may proceed (on a fixed schedule) without a console or any user commands. Output from scheduled jobs is sent via email to one or more administrators defined in the director configuration.

Authentication

Each communication path between Bacula components uses a component name and a password. These passwords are a shared secret, kept as clear text in the Bacula configuration files (although the package installation

methods usually set them to random strings). They are used during authentication to hash the challenge and response tokens. The passwords are never directly sent over the network. It is also possible to configure Bacula with TLS. TLS is discussed in more detail in the section “Advanced Features” later in this chapter.

Configuration

Each Bacula component (except the database server, which is controlled by the database’s configuration files) has its own configuration file consisting of one or more sets of resource definition statements describing Bacula objects and how the system should handle these objects. More complex configurations, such as those with automatically generated components, can be split into multiple files and included individually. These resource definitions can be edited with any standard text editor, although the default configuration supplied with the package represents a sane basic configuration.

Bacula Features

Bacula has a number of features that differentiate it from other open-source backup systems. Here is an overview of these features:

SQL catalog

: The catalog contains a list of files that have been backed up, the basic attributes of each file, including a checksum, the client each file came from, and the volumes where files are stored. Because of this, the catalog is absolutely vital to backups and restores: it stores the records of what has been backed up and when. Although the included `bscan` tool can help reconstruct the catalog should it be damaged, the catalog should, in general, be backed up separately after each backup cycle.

Multivolume backups

: One of Bacula's differentiating features is its excellent native support for multi-volume backups. When equipped with an autochanger, Bacula can simply span tapes without any human intervention (and, if properly configured, can label new volumes on the fly to support this goal). Even on a single-drive machine, Bacula automatically prompts, either via the console or email, for the next tape when it requires additional media.

Flexible media support

: Bacula can back up to different types of media, using disk or tape equally easily. SCSI autochangers simply work "out of the box." Bacula includes tools for easy transfer of backups to CD (`bimagemgr`) or DVD (`dvd-handler`), and the replaceable autochanger support

allows very large storage silos to be integrated with Bacula with a modicum of effort.

Backup levels

As with most other backup systems, Bacula differentiates between full backups (complete dumps), differential backups (files changed since last full backup), and incremental backups (changed files since the last backup of any sort). The desired backup level is specified in the definition or schedule of a backup job, and can be overridden if a job is manually submitted. If a full backup is scheduled, but no full backup is present in the catalog, Bacula automatically promotes a differential or incremental backup job to a full backup job.

Bacula storage format

All data backed up by Bacula is stored on volumes. A volume is simply a repository for backup data; it may be a tape, optical media, or a simple file. Bacula uses a unique format rather than a standard format such as `tar` or `dump`. This design choice ensures consistency across platforms and implementations of Bacula. Consider the use of `tar` for an archive format. Although GNU `tar` is the `tar` present on most Linux implementations, other vendors may use a different implementation, and these implementations are not always compatible with each other, which can then create difficulty unpacking an archive created on a different platform. Bacula avoids this problem by completely specifying the storage format.

While this design choice has distinct advantages, it also means that specialized tools are required to extract data from Bacula backups in the absence of a working Bacula installation. In general, it is best to use a working

Bacula director, catalog, and console to recover files. However, Bacula does provide the `bls` and `bextract` standalone utilities for last-resort recovery when the database is not available. The volume format is thoroughly documented in the Bacula source code and the developer's guide.

The block design of Bacula's storage implementation also allows interleaving of files, which can allow multiple file daemons to communicate with a storage daemon simultaneously, although this slows down file restore by causing more seeks within a storage volume.

Multiple pool support

Backup media in Bacula is drawn from pools, which are essentially lists of available media volumes that can be selected for use. In a small configuration, one pool may be sufficient, but Bacula also supports multiple pools and multiple backup devices extremely well. By supporting multiple pools, Bacula allows easy classification of different machine types or different data handling requirements into different backup pools. An ongoing project will allow migration of job data between pools; this is discussed in the section "Future Directions" later in this chapter.

Macintosh HFS+ support

As of version 1.38.2, Bacula supports resource forks on Macintosh HFS+ volumes; this support allows for correct backup and restoration of files that use resource as well as data forks, and is completely transparent to the user.

Windows VSS support

The Bacula 1.38 Win32 client is Volume Shadow Copy Service-aware; this allows consistent backup of open Windows files, which makes backing up Windows

systems a great deal easier and more reliable. Note that VSS support is present only in Windows XP and Windows Server 2003 and later. There is no VSS support for Windows 95, 98, ME, NT, or 2000: this is a Windows limitation, not a Bacula limitation.

Standalone recovery

Bacula makes it quite easy to create a bootable Linux CD containing everything necessary to bring up a system from bare metal and begin restoration of files. Slightly more difficult (but still efficacious) processes exist for Solaris, FreeBSD, and Windows systems. This is discussed in a bit more detail in the section “Advanced Features” later in this chapter.

Documentation

Documentation is the Achilles’ heel of many open-source projects. Thankfully, this is not the case with Bacula. Although the organization of the manual at the time of writing is a bit lacking, Bacula is thoroughly and accurately documented.

Scalability

Bacula is designed to be an enterprise-class backup system. Because it has clearly defined, logically separated components, it is possible to easily grow a Bacula installation to a very large size indeed. The modular design promotes easy scaling of a site’s backup architecture as it grows: in general, adding capacity is simply a matter of defining additional file daemons or storage daemons and adding them to the director’s configuration. Much work has gone into making Bacula interoperable with many enterprise-class tape silos and label conventions so that it can coexist with large commercial data management and operations.

Despite its suitability for very large networks, Bacula's configuration is simple enough that it is perfectly reasonable to use Bacula to back up a very small network or even just a single machine. Bacula is equally happy backing up 300 clients to a huge tape silo, or one laptop to a weekly burned DVD.

Tape drive testing

One of the most commonly asked questions of Bacula (and probably any other backup system) is "Will it work with my tape drive?" While the online manual does have a good number of drives known to work well, a far more authoritative answer can be obtained using the `btape` utility. This program reads the configuration you have defined for your tape drives and uses it to run a comprehensive series of compatibility tests, including reading, writing, and various seek operations. If `btape` completes all testing successfully, you can be confident that your tape drive is both compatible with Bacula and correctly configured.

An Example Configuration

The best way to illustrate how Bacula is used is to show you an extended example. The example network shown in Figure 6-3 incorporates three machines: an Intel Linux system running the Bacula server components, a Windows XP client, and a Mac OS X client. For simplicity's sake, they are connected to the same network segment, but as long as IP connectivity is available between the server and the clients, this is not a requirement.

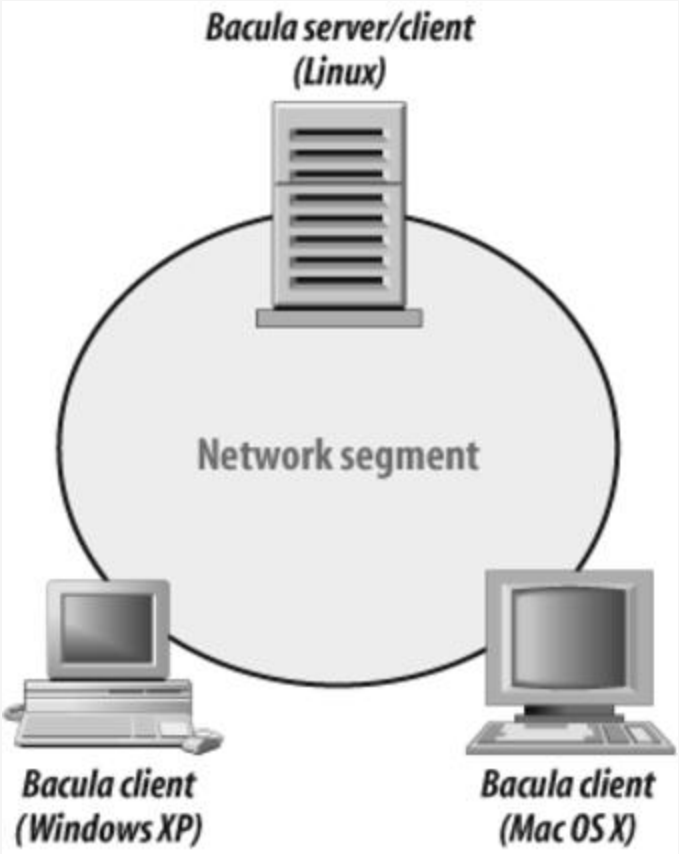


Figure 6-3. Example Bacula configuration

This example configures a recent Bacula distribution (at least version 1.38.2, so that we get Windows VSS support and Mac OS X resource fork support) on the Linux system and configures the Linux server to back up some of its own

files. Once we have that working, we'll install and configure the Windows client and the Macintosh client and back up directories on each of them. In our sample configuration, we'll back up */etc* on the Linux system, using a directory on the server as our storage device rather than tape or optical media.

Setting Up the Server

To install the Bacula application, we'll use RPMs provided by http://www.bacula.org, which can be installed using standard Linux system management techniques and tools. The packages install the Bacula configuration files for Linux in */etc/bacula*, as mandated by the Linux Standard Base (LSB). For ease of installation, we'll use the SQLite database (in a production network, you would use MySQL or PostgreSQL).

> **Tip ✎**
>
> Bacula is also available in Debian and other packaging formats. Although the version of Bacula available in Debian Sarge is ancient, Debian Etch is tracking recent Bacula releases, and backports of new Bacula versions to Sarge exist.

For our simple example, just rename the default client ID of `Client1` to `test1`, and define a `FileSet` resource using the following line:

```
File = /etc
```

> **Tip ✎**
>
> A `FileSet` specifies files to back up for a given `Job`.

The following line is needed in the default pool definition to tell Bacula to create new sequentially numbered volumes as it requires them:

```
Label Format = TestVol
```

The other supplied defaults are suitable for most small configurations.

Configuring the storage daemon is just as easy; again, relying on Bacula to supply sane default settings, use the `FileStorage` storage resource, pointing it to */opt/backups* (owned by the *bacula* user) rather than to */tmp*; ensure that `Label Media = yes` is set; and make sure that the passwords agree for the director, file daemon, and storage daemon. Following these updates, restart the Bacula services using the */etc/init.d* scripts installed with the package, and run the Bacula `bconsole` to test the installation.

Initial Backup (Linux Client)

Once the Bacula console initializes, use the supplied job definitions to initiate a backup job (`test1` is the job we created with the settings shown earlier):

```
*run
A job name must be specified.
The defined Job resources are:
  1: test1
  2: BackupCatalog
  3: RestoreFiles
Select Job resource (1-3): 1
Run Backup job
JobName: test1
FileSet: Etc
Level: Incremental
Client: test1-fd
Storage: File
Pool:  Default
When:  2006-01-20 09:30:38
Priority: 10
```

```
OK to run? (yes/mod/no): yes
Job started. JobId=4
```

After a while, the console says:

```
You have messages.
```

Here are the messages:

```
*messages
[ ... lines detailing automatic creation/labeling of volume omitted ... ]
 JobId:     4
 Job:     test1.2006-01-20_09.30.39
 Backup Level:   Full (upgraded from Incremental)
 Client:     "test1-fd" i686-redhat-linux-gnu,redhat,(Heidelberg)
 FileSet:    "Etc" 2006-01-19 10:47:39
 Pool:     "Default"
 Storage:    "File"
[ ... lines omitted ... ]
 FD Bytes Written:  40,129,258
 SD Bytes Written:  40,656,235
[ ... lines omitted ... ]
Termination:   Backup OK
```

This response shows that the backup succeeded and backed up about 40 MB. A full backup was actually performed even though we specified an incremental backup (because no full backup had been performed yet).

“I Can’t Afford Backups”

My friend Greg is a special-effects guru who does modeling/texturing, animation, and ultra-hi-resolution photography. He typically works on images that are a gigapixel in size, where merely saving the file on his computer can take up to three hours. He has purchased several terabytes of disk to store all his different projects over the years but maintains no backups. He has no machine room and no special cooling or storage system for his components. They all sit on a dusty wood floor. Late last year his striped terabyte RAID array crashed, and he lost five years’ worth of work. Now his main goal is to save enough money to get the disk repaired by

DriveSavers. In the meanwhile, he's buying other disks and putting new work on them—again without any backups. Greg says that he can't afford a backup system that could handle the truly massive amounts of data he regularly uses, and basically hopes that if he gets new, "more reliable" drives, and if he listens closely for sounds that tell when the drive is being weird, then he'll be in good shape again—at least until the next catastrophic crash happens.

Jason

Initial Restore (Linux Client)

To test the backup, we'll restore a file. Choosing the file */etc/protocols* at random, use `bconsole`'s `restore` command to initiate the restore. From `bconsole`, enter the `restore` command, select option 7 to enter a list of files, select a client, enter the file to be restored, and press Enter.

Now, we'll verify that the file *protocols* is present in */tmp/bacula-restores/etc/protocols*. Although it is possible to restore files in place, Bacula's default configuration restores them under */tmp/bacula-restores* to allow you to verify that the restored files are correct before committing them to their actual locations in the filesystem.

Windows Backup

Next let's back up some files on the Windows workstation. First, use a text editor to define a new `Job`, `FileSet`, and `Client` in *bacula-dir.conf*:

```
Job {
 Name = "WinXP-test"
 Client = winxp-fd
 JobDefs = "DefaultJob"
```

```
 FileSet = "WinTest"
}
FileSet {
 Name = "WinTest"
 Enable VSS = yes
 Include {
[ ... lines omitted ... ]
 File = C:/Windows/SYSTEM32/DRIVERS/ETC
 File = "C:/Program Files/Wizards/eTools/User"
 }
[ ... lines omitted ... ]
}
# Client (File Services) to backup
Client {
 Name = winxp-fd
 Address = vpc-xp2.fsf.net
...
}
```

The important points to notice here are `Enable VSS = yes` in the `FileSet` resource, which turns on VSS in Windows, and the format of the `FileSet` resource. Windows files are specified with forward slashes, and, if the filename contains spaces, it must be enclosed in double quotes. Use the `reload` command in the console to enable the director to pick up changes to its configuration files.

Next, install the Bacula Windows client on the Windows workstation. It is a standard Windows package, installed like any other Windows application. The configuration files require editing to reflect the director (`test1-dir`) and its password.

Once installed, the file daemon runs as a Windows service, and a little icon of a cassette tape appears in the Windows tray as described earlier. Running the `wx-console` command in the Bacula install directory displays a screen like that shown in Figure 6-4.

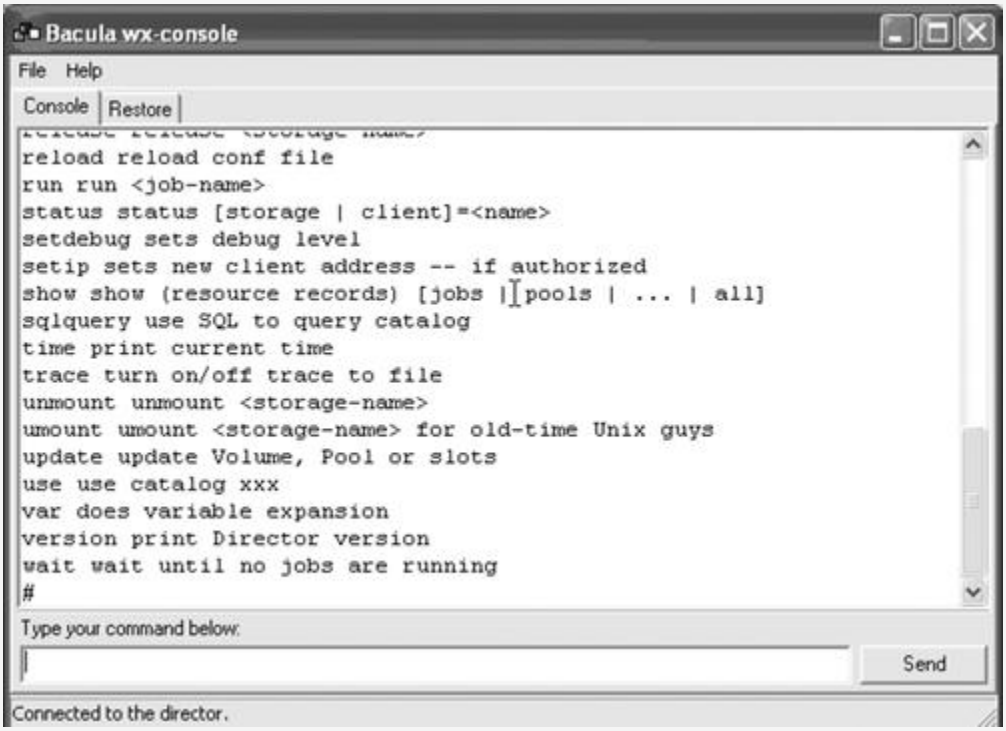


Figure 6-4. wx-console

Running a backup job is very similar to running one for the Linux client, except that we'll run the `WinXP-test Job` rather than `test1`. The director logs look almost identical to the earlier backup:

```
20-Jan 14:49 winxp-fd: Generate VSS snapshots. Driver="VSS
WinXP", Drive(s)="C"
[ ... lines omitted ... ]
Termination:   Backup OK
```

Let's try something a bit different for the file restore, restoring this file:

```
C:\Program Files\Wizards\eTools\User\Characters\lola13.chr
```

Rather than enter a list of files, let's choose the option to restore the most recent backup for our Windows client, and use `bconsole` rather than `wx-console` for this job. `bconsole` shows a directory listing interface; navigate with `cd` and choose the file to restore with the `mark` command (with `wx-console`, you navigate the directory tree with the mouse and click on filenames to mark them).

Running the restore job once again puts the file under */tmp/bacula-restores*, which is translated to *C:\tmp\bacula-restores* in the Windows filesystem. From there, it can be copied back into place.

Mac OS X Backup

Mac OS X is really BSD Unix under the hood, so it should not be surprising that Bacula behaves on it very much as it does on any other Unix or Linux machine. Configuring the director is basically the same as adding the Windows client. However, in the `FileSet` resource, you add this directive:

```
HFS Plus Support = yes
```

This option, which enables proper backing up of resource forks, should always be specified for Mac OS X clients.

For this example, let’s specify the following directory to back up:

```
File = "/Users/adam/Documents/Emulators/Virtual II"
```

> **Tip ✎**
>
> Note that because the filename contains a space, the name must be enclosed in quotation marks.

We chose this directory because it contains an application bundle. Macintosh applications are actually directory structures containing not only the binary executable, but resources and support files. If the application is to run successfully, the entire structure, including any associated resource forks, must be maintained.

The backup process is identical to backing up any other client, and its output looks exactly the same. After backing

up that directory, remove it entirely, deleting the Virtual II application.

When we run the restore job this time, we'll use the `restore all` command; this is equivalent to marking all files in the tree in our Windows example (but obviously a great deal faster). As a result, the restore job restores the entire directory under */tmp/bacula-restores*. After moving the directory back to its correct location, the Virtual II application runs without incident.

This example demonstrates backup and restore for three different clients: Linux, Windows XP, and Mac OS X. All three are very similar in use; the primary difference in command usage is not system-specific, but simply has to do with which restore option is chosen: single named file restore, marking files in a filesystem tree, or restoring an entire tree. On the server side, we need a single line of configuration for each non-Linux client to preserve unique filesystem features: VSS support for Windows and HFS+ support for Mac OS X.

The default method of file restoration is to restore the files under */tmp/bacula-restores* (the default) and then copy them to their final destinations once you are sure that you have restored the right files in their correct versions. Once familiar with Bacula's operation, many users choose to restore files directly to their original locations.

Advanced Features

In addition to being able to back up and restore files over the network, Bacula has many advanced features, a brief summary of which follows. Complete information on these features can be found in the Bacula documentation at http://www.bacula.org.

Bare-Metal Recovery

The advanced feature of greatest importance to most users is the ability to do bare-metal recovery of machines using Bacula. Bacula provides tools for creating customized bootable CD images that can be used for recovery. This support is fairly complete for Linux. It is a largely manual process on Solaris and FreeBSD. A very different, but fairly efficient, process exists for Windows recovery. At this time, bare-metal restore is not possible for Macintosh clients.

Bacula can be run from a bootable rescue CD-ROM for Linux. This CD-ROM contains a minimal copy of the machine's OS, a statically linked Bacula file daemon, and configuration files describing the machine for which the rescue CD-ROM was created.

The basic strategy of the rescue CD is to boot, repartition the hard drive as described on the CD-ROM, bring the machine back onto the network, and then restore the files for that machine back onto the newly formatted drive. The process of building the CD-ROM creates scripts that handle network configuration, disk partitioning, and writing the appropriate boot record (using `grub` or `lilo`). A single CD-ROM can be used for multiple clients as long as each client has its own configuration directory on the CD-ROM.

As a last resort, if you have a suitable rescue disk and have backed up all the rescue files generated by the CD creation script, you should be able to create a usable Bacula restore CD even if the client has been lost.

Solaris and FreeBSD follow similar patterns, although without the benefit of a dedicated recovery CD: boot from rescue or installation media, format the disks appropriately, bring up the network, install a statically linked file daemon, and then restore all backup files. Rudimentary automated script creation support exists on Solaris, although not to the same degree as it does on Linux.

Because building the rescue CD-ROM requires a separate manual process for each client, some Linux users may find it easier to use Knoppix instead (see Chapter 11). Knoppix provides a run-from-CD Linux distribution that includes `bacula-fd`. If Knoppix is used, setting up the network and then partitioning and formatting the disk is a manual process, but once the user has done that, the restoration proceeds precisely as it does for any other Bacula restore job. The boot record creation must also be done by hand in this case.

Windows relies on the user having run `ntbackup` to save a copy of critical Windows system files prior to the Bacula backup (see Chapter 3); this can be implemented with client-side hooks, as described later in this section. In this case, install a minimal Windows system first, then install the Bacula client and restore the backup including the output of `ntbackup`, and finally restore the system identity from the `ntbackup` image.

Windows XP users have it a little easier: they can create a BartPE rescue CD that includes a Bacula client. PEBuilder, the program that generates this image can be found at http://www.nu2.nu/pebuilder/; it assists in creating a bootable Windows rescue CD. PEBuilder requires access to

Windows installation media to supply the necessary Windows files.

Backup Traffic and Storage Encryption

Frequently, you have to back up machines over a network that can't be trusted. The most obvious case is backup over a WAN (or the Internet). Corporate compliance regulations may specify that such traffic must be encrypted in transit.

It has always been possible to tunnel Bacula using `stunnel` or `ssh` port-forwarding. This requires no Bacula configuration support other than pointing the various daemons at the proper ports; authentication issues are handled by `stunnel` or `ssh`.

In recent versions, Bacula includes TLS support. With TLS support, you can ensure that traffic between the file daemon, the director, and the storage daemon is not transmitted in clear text. Additionally, TLS provides another layer of authentication on top of the requirement for matching passwords in the daemons.

Bacula can use self-signed certificates created with a locally deployed *certificate authority* (CA). The Bacula manual discusses sources of information about creating a local CA. All the usual caveats about self-signed certificates apply to Bacula. However, the user does not know whether the certificate has been signed by a trusted CA, so as long as the system administrator trusts his own ability to sign and deploy certificates, there is little benefit to paying for a signed certificate for an internal backup infrastructure. If backup is being offered as a managed service, on the other hand, it may well be worth the trouble and expense to acquire a properly signed SSL certificate for the backup service.

Bacula 1.39 and later supports encryption of the actual stored data to prevent access to backup contents by unauthorized personnel. This is in addition to the traffic between Bacula components. Large organizations are beginning to require that backup data be encrypted when transferred and stored. However, it's equally useful for the casual user backing up to DVD-R, who would prefer that the thief who steals his backpack containing the backup DVD not also get all his personal data.

The encryption code uses TLS certificates to manage the encryption keys that secure the data. This allows you to optionally specify multiple master decryption keys for escrow purposes, helping to protect against data loss in case a single private key is lost. Encryption can be enabled on a client-by-client basis.

Python Script Support

Bacula includes support to run a script before or after a given backup job. This allows a job to invoke an external program, which can then communicate with Bacula via its standard output. Although not a particularly sophisticated tool, this can be effective for ensuring that all buffers are flushed before a backup run begins.

Recent versions of Bacula also support an embedded Python interpreter in the director, file daemon, and storage daemon. A Python script, running inside the Bacula director's address space, can register to handle any of a number of actions. One common use for this is to change job parameters on the fly in reaction to the environment: for instance, if a new volume name is required, the Python support can be employed to create it at runtime, perhaps incrementing a counter to generate the logical next volume name in a series. The embedded Python interpreter has full

knowledge of Bacula's internal state; this gives it much more flexibility than the external-program approach, in which only a few pieces of information can be passed as parameters to the external program. An embedded interpreter in the file daemon allows for very sophisticated client-side job pre- and post-processing.

Client Script Support

Bacula allows the user to specify, as part of a job, a program that runs on the client before or after the backup occurs. This is particularly useful for backing up database servers. For example, the user may request that the database create (using its own maintenance tools) a flat-file snapshot of its state just before the job executes. After the backup is finished, the file dump can be removed to conserve space on the client.

Another common use of client-side hooks is to use `ntbackup` to dump essential Windows configurations to a flat file that Bacula can restore.

Autochanger Support

Bacula interacts quite well with SCSI tape autochangers; in general, it uses the `mtx` command to manipulate the autochanger but wraps this command inside Bacula's own replaceable `mtx-changer` script. You can edit this script to customize `mtx` in your environment.

The autochanger can label new volumes it finds, creating volume names based on a pattern, a barcode reader, or a Python script, and automatically mounting them. For large backups, where a single job spans multiple tapes, autochanger support makes it possible for Bacula to use as many tapes as it needs without requiring human

intervention. It also automates the restore process because Bacula can request the correct tape from the autochanger in order to retrieve the volume with the desired backups.

For most purposes, the supplied script is adequate and contains hints for small site-specific customizations. For instance, it can easily be modified to simulate label barcodes for autochangers that do not support barcode tape identification.

However, `mtx-changer` is a much more generally applicable solution; it simply defines Bacula's interface to the autochanger and therefore can be replaced in its entirety. This allows Bacula to interact with devices that are radically different from SCSI autochangers, with no changes required to the core Bacula code.

One example of this is support for mainframe tape silos: in these environments, `mtx-changer` has been replaced with a network client and server so that Bacula can request tapes from a tape management system running under z/VM, despite the fact that the tape silo does not have anything like a SCSI autochanger interface.

ANSI and IBM Tape Labels

Bacula, as of version 1.38, can use standard ANSI and IBM tape labels. This enables Bacula to coexist peacefully with other enterprise backup systems and share a unified tape catalog with them.

File-Based Intrusion Detection

One of the simplest ways to monitor the status of a host is to watch for filesystem changes. Bacula is already watching the filesystem and storing information about it in the catalog, including filenames, sizes, permissions, and

checksums. By periodically running a job of `type verify` against the catalog, you can use Bacula as a simple intrusion detection system, along the lines of `tripwire`.

Future Directions

Bacula's major long-term goal is to become a competitive enterprise-class backup system. Some enhancements, particularly to search speed in large catalogs, are necessary before it will be on the same footing with commercial enterprise backup systems.

Usability enhancements are also vital: chief among them is construction of a more complete and maintainable GUI for Bacula.

The list of in-progress Bacula projects can be found at http://bacula.cvs.sourceforge.net/bacula/bacula/projects?view=markup and is updated every six months or so. As of this writing, the top five unfinished projects are those detailed in the following sections (by the time this book is published, some or all of these may be present in the mainline Bacula code).

Pool Migration

This project aims to implement migration of stored data from one pool to another, which allows implementation of tiered storage; for example, backups can initially occur to disk, and then migrate to other storage media on a controlled schedule.

Migration of stored data has two benefits. First, by allowing optional use of the target media, pool migration can better use storage media. That is, it acts as a sophisticated spooling facility, which is especially important when using write-once media; in that case, it's best to use as much capacity on each volume as possible.

The second is convenience: although tape or optical storage may be the desired long-term storage medium, the

most common use of Bacula is restoration of a single recently deleted file or directory. It is much more convenient if recent backups are maintained on disk so that the user doesn’t have to search for backup media. However, the amount of disk space required can be kept reasonably small because backups are automatically migrated onto archival media according to a scheduling policy.

Copy pool creation is a related feature (it is the eighth item on the list), and is designed with off-site archival in mind. A single job writes backup output to volumes in multiple pools simultaneously; this facilitates, for instance, the creation of media for off-site storage.

As of November 2006, pool migration is functional, though still slightly rough around the edges. By the time of publication, it should be a stable and fully supported feature.

Tracking Deleted/Renamed Files

When Bacula does a differential or incremental backup, it does not notice whether a file or directory has vanished, only whether its contents have changed. Thus restoration of a file set from a particular date, which requires first a full restoration and then application of a differential or incremental restoration, restores files that were present in the full backup but had actually been deleted or renamed by the time the differential or incremental backup was taken. A similar situation exists with respect to files whose hard link count has changed. Note that as it stands, no data is lost by Bacula; however, until this item is implemented, unwanted data may be restored and have to be deleted manually. The primary use of this item is creation of a correct point-in-time snapshot.

Python-Based GUI Tool

The existing wxWidgets GUI is written in C++ and is difficult to maintain. An enhanced Python-based GUI built with Qt Designer will be easier to maintain, more accessible to user customization, and more easily portable.

Base Job Support

A “base” job is one that backs up files that are expected to change very rarely and are likely to be identical across many clients. If you have 50 Linux machines all running the same distribution and release, most of the system binaries are both identical across all those machines and are unlikely to change often. A single job that could represent the initial shared state of these machines would vastly reduce the amount of backup media required as well as the amount of media changing required if mass recovery of all these machines were ever needed.

Client-Initiated Backups

Consider an organization with intermittently connected machines, such as a company with many users who often travel with their laptops, and who may not be in the office on any given day. It would be very helpful for such an organization to allow the client to initiate a backup when it recognizes that it is on its home network and can connect to the Bacula director.

Plug-in Support for File Daemons

This project is lower down on the list than the previous five projects, but it represents an extremely interesting

possibility for creating a general structure for Bacula backup of challenging applications (e.g., running databases). While the existing file daemon supports backing up anything that can be stored as a file, the fact that Bacula does all work over TCP/IP opens up the possibility of more special-purpose file daemon plug-ins designed to back up more difficult applications. For example, at this time, at least one user has reported success in creating a simple, Windows-based file daemon plug-in that, instead of backing up files, knows how to interact with MS SQL server to perform full, proper backups. Work to use this technique to integrate Bacula with VM/CMS and with AFS is also underway.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 7. Open-Source Near-CDP

Chapter 8 discusses commercial backup software and the concept of *near-continu ous data protection* (near-CDP) systems, which is basically replication tied with some type of snapshot product. Before moving to the commercial side of things, this chapter examines three open-source near-CDP systems. The first section covers `rsync` with snapshots and explains the concept, originally popularized by Mike Rubel. The `rsnapshot` section explains an open-source project based on that concept. It's designed to automate the backup of user files and does not work very well with databases and other large files. Finally, the `rdiff-backup` section explains a different project with a number of features, including advanced metadata capabilities and an ability to handle large files.

> **Tip ✎**
>
> This chapter was contributed by Michael Rubel, David Cantrell, and Ben Escoto. Mike is a graduate student in aeronautics at Caltech, where he keeps several backups of his thesis (which he hopes to finish soon). David doesn't believe in quiche. Ben is currently an actuarial analyst at Aon Re.

Replication by itself wouldn't work. You can't just have one system `rsync` to another system as a backup because logical corruption (such as a virus or user error) would be replicated to the backup. You need to have some way of keeping multiple versions of data on the target so that you can go back to a previous version when necessary.

If you can do this, you can have a continuously incremental system that always has a full backup of any version

available. This requires, of course, enough space to store one full copy plus incremental backups. It can even be employed in a variety of ways: push-to-server (as just described), pull-from-source, or locally (for example, to a removable hard disk drive).

rsync with Snapshots

The idea of using replication to create a snapshot-like system was popularized by Mike Rubel in an online article about `rsync` with snapshots (http://www.mikerubel.org/computers/rsync_snapshots/). The idea is both simple and powerful.

You first have to copy one directory to another directory. To do this, copy the source directory to a target directory (`cp -a /home /backups/home.0`). Because you will need them later, the target directory must support *hard links*. (Hard links are covered in more detail later in this chapter.)

Next you need to make another "copy" of the data you backed up; this copy is used as the previous version when you update this version. To do this, copy the target directory to a .<*n*> version, where *n* is any digit. If you do this using hard links, the second "copy" won't take up any space, so use the `-l` option of `cp` to tell it to use hard links when it makes the copy (`cp -al /backups/home.0 /backups/home.1`). Now there are two identical copies of our source directory on our backup system (*/ backups/home.0* and */backups/home.1*) that take up the size of only one copy.

Now that you've copied the backup to another location, it's time to make another backup. To do this, identify any files in the source that are new or changed, remove them in the target directory if they're there, then copy them to the target directory. If it's an updated version of a file already in our target directory, it must be unlinked first. You can

use the `rsync` command to do this all in one step (`rsync --delete –av / home/. /backups/home.0/`). This step is the heart of the idea. By removing a file that was already in the backup directory (*/backups/home.0*), you sever the hard link to the previous version (*/backups/home.1*). But since you used hard links, the previous version is still there (in */backups/home.1*), and the newer version is in the current backup directory (in */backups/home.0*).

The Reason We Do Backups

When working at a university, I had a grad student come to me and ask me to restore the contents of her home directory. No problem. I restored it, and she went away.

The next day, she came back and asked for it to be restored again. I told her to be more careful, and we aliased `rm` to `rm -i` for her.

The next day, she was back again. Curious as to how this kept happening, I checked the *.history* file:

```
$ echo y | rm *
```

We aliased `rm` to `echo no` after that.

Brian O'Neill

To make another backup, move the older directory (*/backups/home.1*) to an older number (*/backups/home.2*), then repeat the hard-link copy, unlink, and copy process. You can do this as many times as you want and keep as many versions as you want. The space requirements are modest; the only increase in storage is for files that are new with that backup.

An Example

Let’s back up the directory */home*. The example directory has three files:

```
$ echo "original myfile" >/home/myfile.txt
$ echo "original myotherfile" >/home/myotherfile.txt
$ echo "original mythirdfile" >/home/mythirdfile.txt
$ ls -l /home
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt
```

Now let’s create a copy of */home* in */backups/home*.

```
$ cp -a /home /backups/home.0
$ ls -l /backups/home.0
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du -sb /backups
58      /backups
```

Note that each file shows a 1 in the links column, there is one copy of the */home* directory in */backups* and it contains the same files as */home*, and the entire */backups* directory takes up 58 bytes, which is the same as the number of bytes in all three files. Now let’s create a second copy using hard links:

```
$ cp -al /backups/home.0 /backups/home.1
$ ls -l /backups/*
/backups/home.0:
total 3
-rw-r--r--  2 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.1:
total 3
```

```
-rw-r--r--  2 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du -sb /backups
58      /backups
```

Now you can see that there are two copies of */home* in */backups*, each contains the same files as */home*, and they still take up only 58 bytes because we used hard links. You should also note that the links column in the `ls -l` listing now contains a 2. Now let's change a file in the source directory:

```
$ ls -l /home/myfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35
/home/myfile.txt
$ echo "LET'S CHANGE MYFILE" >/home/myfile.txt
$ ls -l /home/myfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 20 Jul  8 19:41
/home/myfile.txt
```

Please note that the size and modification time of *myfile.txt* changed. Now it's time to make a backup. The process we described earlier would notice that */home/myfile.txt* has changed and that it should be removed from the backup directory and copied from the source. So let's do that:

```
$ rm /backups/home.0/myfile.txt
$ cp -a /home/myfile.txt /backups/home.0/myfile.txt
$ ls -l /backups/*
/backups/home.0:
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.1:
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
```

```
myotherfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du –sb /backups
78
```

Now you can see that *myfile.txt* in */backups/home.1* has only one link (because we removed the second link in */backups/home.0*), but it still has the same size and modification date as before. You can also see that */backups/home.0* now contains the new version of *myfile.txt*. And, perhaps most importantly, the size of */backups* is now the original size (58 bytes) plus the size of the new version of *myfile.txt* (20 bytes), for a total of 78 bytes. Now let's get ready to make another backup. First, we have to create the older versions by moving directories around:

```
$ mv /backups/home.1 /backups/home.2
$mv /backups/home.0 /backups/home.1
```

Then we need to create the new previous version using `cp -al`:

```
$ cp -al /backups/home.1 /backups/home.0
$ ls -l /backups/*
/backups/home.0:
total 3
-rw-r--r--  2 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.1:
total 3
-rw-r--r--  2 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.2:
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
```

```
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du -sb /backups
78      /backups
```

Now we have */backups/home.2*, which contains the oldest version, and */backups/home.1* and */backups/home.0*, which both contain the current backup. Please note that the size of */backups* hasn't changed since the last time we looked at it; it's still 78 bytes. Let's change another file and back it up:

```
$ echo "LET'S CHANGE MYOTHERFILE" >/home/myotherfile.txt
$ rm /backups/home.0/myotherfile.txt
$ cp -a /home/myotherfile.txt /backups/home.0/myotherfile.txt
$ ls -l /backups/*
/backups/home.0:
total 3
-rw-r--r--  2 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 25 Jul  8 19:45
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.1:
total 3
-rw-r--r--  2 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.2:
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du -sb /backups
103     /backups
```

You can see that */backups/home.0* now contains a different version of *myotherfile.txt* than what is in the other directories, and that the size of */backups* has changed from 78 to 103, which is a difference of 25—the size of the new version of *myotherfile.txt*. Let's prepare for one more backup:

```
$ mv /backups/home.2 /backups/home.3
$ mv /backups/home.1 /backups/home.2
$ mv /backups/home.0 /backups/home.1
$cp -al /backups/home.1 /backups/home.0
```

Now we'll change one final file and back it up:

```
$ echo "NOW LET'S CHANGE MYTHIRDFILE" >/home/mythirdfile.txt
$ rm /backups/home.0/mythirdfile.txt
$ cp -a /home/mythirdfile.txt /backups/home.0/mythirdfile.txt
$ ls -l /backups/*
/backups/home.0:
total 3
-rw-r--r--  3 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 25 Jul  8 19:45
myotherfile.txt
-rw-r--r--  1 cpreston mkgroup-l-d 29 Jul  8 19:51
mythirdfile.txt

/backups/home.1:
total 3
-rw-r--r--  3 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 25 Jul  8 19:45
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.2:
total 3
-rw-r--r--  3 cpreston mkgroup-l-d 20 Jul  8 19:41 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

/backups/home.3:
total 3
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 myfile.txt
-rw-r--r--  2 cpreston mkgroup-l-d 21 Jul  8 19:35
myotherfile.txt
```

```
-rw-r--r--  3 cpreston mkgroup-l-d 21 Jul  8 19:35
mythirdfile.txt

$ du –sb /backups
132
```

Again, the total size of */backups* has changed from 103 bytes to 132, a difference of 29 bytes, which is the size of the new version of *mythirdfile.txt*.

The proof is in the restore, right? Let's look at all versions of all files by running the `cat` command against them:

```
$ cat /backups/home.3/*
original myfile
original myotherfile
original mythirdfile
$ cat /backups/home.2/*
LET'S CHANGE MYFILE
original myotherfile
original mythirdfile
$ cat /backups/home.1/*
LET'S CHANGE MYFILE
LET'S CHANGE MYOTHERFILE
original mythirdfile
$ cat /backups/home.0/*
LET'S CHANGE MYFILE
LET'S CHANGE MYOTHERFILE
NOW LET'S CHANGE MYTHIRDFILE
```

You can see that the oldest version (*/backups/home.3*) has the original version of every file, the next newest directory has the modified *myfile.txt*, the next newest version has that file and the modified *myotherfile.txt*, and the most recent version has all of the changed versions of every file. Isn't it a thing of beauty?

Beyond the Example

The example, while it works, is simple to understand but kind of clunky to implement. Some of the steps are rather manual, including creating the hard-linked directory, identifying the new files, removing the files you're about to

overwrite, and then sending the new files. Finally, our manual example does not deal with files that have been deleted from the original; they would remain in the backup. What if you had a command that could do all that in one step? You do! It's called `rsync`.

The `rsync` utility is a very well-known piece of GPL'd software, written originally by Andrew Tridgell and Paul Mackerras. If you have a common Unix variant, you probably already have it installed; if not, you can install a precompiled package for your system or download the source code from rsync.samba.org and build it yourself. `rsync`'s specialty is efficiently synchronizing file trees across a network, but it works well on a single machine, too. Here is an example to illustrate basic operation.

Suppose you have a directory called *<source>/* whose contents you wish to copy into another directory called *<destination>/*. If you have GNU `cp`, you can make the copy like this:

```
$ cp -a source/. destination/
```

The archive flag (`-a`) causes `cp` to descend recursively through the file tree and to preserve file metadata, such as ownerships, permissions, and timestamps. The preceding command first creates the destination directory if necessary.

Tip ✎

It is important to use *<source>/.* and not *<source>/** because the latter silently ignores top-level files and subdirectories whose names start with a period (.). Such files are considered hidden and are not normally displayed in directory listings, but they may be important to you!

However, if you make regular backups from *<source>/* to *<destination>/*, running `cp` every time is not efficient because even files that have not changed (which is most of them) must be copied every time. Also, you would have to periodically delete *<destination>/* and start fresh, or backups of files that have been deleted from *<source>/* will begin to accumulate.

Fortunately, where `cp` falls short at copying mostly unchanged filesystems, `rsync` excels. The `rsync` command works similarly to `cp` but uses a very clever algorithm that copies only changes. The equivalent `rsync` command would be:

```
$ rsync -a source/. destination/
```

> **Warning ☠**
>
> `rsync` is persnickety about trailing slashes on the source argument; it treats *<source>* and *<source>/* differently. Using the trailing */.* is a good way to avoid ambiguity.

You'll probably want to add the `--delete` flag, which, in addition to copying new changes, also deletes any files in *<destination>/* that are absent (because they have presumably been deleted) from *<source>/*. You can also add the verbose flag (`-v`) to get detailed information about the transfer. The following command is a good way to regularly synchronize *<source>/* to *<destination>/*:

```
$ rsync -av --delete source/. destination/
```

`rsync` is good for local file synchronization, but where it really stands out is for synchronizing files over a network. `rsync`'s unique ability to copy only changes makes it very fast, and it can operate transparently over an `ssh`

connection. To `rsync` from */<source>/* on the remote computer *example.oreilly.com* to the local directory */<destination>/* over `ssh`, you can use the command:

```
$ rsync -av --delete username@example.oreilly.com:/source/. /destination/
```

That was pull mode. `rsync` works just as well in push mode from a local */<source>/* to a remote */<destination>/:*

```
$ rsync -av --delete /source/. username@example.oreilly.com:/destination/
```

As a final note, `rsync` provides a variety of `--include` and `--exclude` options that allow for fine-grained control over which parts of the source directory to copy. If you wish to exclude certain files from the backup—for example, any file ending in *.bak* or certain subdirectories—they may be helpful to you. For details and examples, see the `rsync` manpage.

Understanding Hard Links

Hard links are an important Unix concept to understand if you're going to use this technique. Every object in a Unix filesystem, which includes every directory, symbolic link, named pipe, and device node, is identified by a unique positive integer known as an *inode number*. An inode keeps track of such mundane details as what kind of object it is, where its data lives, when it was last updated, and who has permission to access it. If you use `ls` to list files, you can see inode numbers by adding the `-i` flag:

```
$ ls -i foo
409736 foo
```

What does *foo* consist of? It consists of an inode (specifically, inode number 409736) and some data. Also—and this is the critical part—there is now a record of the new inode in the directory where *foo* resides. It now lists the name *foo* next to the number 409736. That last part, the

entry of a name and inode number in the parent directory, constitutes a *hard link*.

Most ordinary files are referenced by only one hard link, but it is possible for an inode to be referenced more than once. Inodes also keep a count of the number of hard links pointing to them. The `ls –l` command has a column showing you how many links a given file has. (The number to the left of the file owner's user ID is the number of hard links—one in the following example.)

```
$ ls -l foo
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 foo
```

To make a second link to a file within a filesystem, use the `ln` command. For example:

```
$ ln foo bar
$ ls -i foo bar
409736 foo
409736 bar
```

> **Tip ✎**
>
> Hard links, like the one illustrated here, can be created only for files within the same filesystem.

Now the names *foo* and *bar* refer to the same file. If you edit *foo*, you are simultaneously editing *bar*, and vice versa. If you change the permissions or ownership on one, you've changed them on the other, too. There is no way to tell which name came first. They are equivalent.

When you remove a file, all you're removing is a link to that inode; this is called *unlinking*. An inode is not actually released until the number of links to it drops to zero. *(*Even then, the inode is freed only when the last process has finished using it, which is why you can remove the executable of a running program.) For example, `ls -l` now

tells you that *foo* has two links. If you remove *bar*, only one remains:

```
$ ls -l foo
-rw-r--r--  2 cpreston mkgroup-l-d 16 Jul  8 19:35 foo
$ rm bar
$ ls -l foo
-rw-r--r--  1 cpreston mkgroup-l-d 16 Jul  8 19:35 foo
```

The situation would have been the same if you'd removed *foo* and run `ls -l` on *bar* instead. If you now remove *foo*, the link count drops to zero, and the operating system releases inode number 409736.

Here is a summary of some of the important properties of hard links. If *foo* and *bar* are hard links to the same inode:

- Changes to *foo* immediately affect *bar*, and vice versa.
- Changes to the metadata of *foo*—the permissions, ownership, or timestamps—affect those of *bar* as well, and vice versa.
- The contents of the file are stored only once. The `ln` command does not appreciably increase disk usage.
- The hard links *foo* and *bar* must reside on the same filesystem. You cannot create a hard link in one filesystem to an inode in another because inode numbers are unique only within filesystems.
- You must unlink both *foo* and *bar* (using `rm`) before the inode and data are released to the operating system.

Hard-Link Copies

In the previous section, you learned that `ln foo bar` creates a second hard link called *bar* to the inode of file *foo*. In many respects, *bar* looks like a copy of *foo* created at the same time. The differences become relevant only when you try to

change one of them, examine inode numbers, or check disk space. In other words, as long as in-place changes are prohibited, the outcomes of `cp foo bar` and `ln foo bar` are virtually indistinguishable to users. The latter, however, does not use additional disk space.

Suppose you wanted to make regular backups of a directory called *<source>/*. You might make the first backup, a full copy, using `rsync`:

```
$ rsync -av --delete source/. backup.0/
```

To make the second backup, you could simply make another full copy, like so:

```
$ mv backup.0 backup.1
$ rsync -av --deletesource/. backup.0/
```

That would be inefficient if only a few of the files in *<source>/* changed in the interim. Therefore, you create a second copy of *backup.0* to use as a destination (since you're using hard links, it won't take up any more space). You can use two different techniques to make backups in this way. The first is a bit easier to understand, and it's what we used in the example. The second streamlines things a bit, doing everything in one command.

GNU `cp` provides a flag, `-l`, to make hard-link copies rather than regular copies. It can even be invoked recursively on directories:

```
$ mv backup.0 backup.1
$ cp -al backup.1/. backup.0/
$ rsync -av --delete source/. backup.0/
```

Putting `cp -al` in between the two backup commands creates a hard-linked copy of the most recent backup. You then `rsync` new changes from *<source>/* to the *backup.0*. `rsync` ignores files that have not changed, so it leaves the links of unchanged files intact. When it needs to change a file, it unlinks the original first, so its partner is unaffected.

As mentioned before, the `--delete` flag also deletes any files in the destination that are no longer present in the source (which only unlinks them in the current backup directory).

`rsync` is now taking care of a lot. It decides which files to copy, unlinks them from the destination, copies the changed files, and deletes any files it needs to. Now let's take a look at how it can handle the hard links as well. `rsync` now provides a new option, `--link-dest`, that will do this for you, even when only metadata has changed. Rather than running separate `cp -al` and `rsync` stages, the `--link-dest` flag instructs `rsync` to do the whole job, copying changes into the new directory, and making hard links where possible for unchanged files. It is significantly faster, too.

```
$ mv backup.0 backup.1
$rsync -av --delete --link-dest=../home.0 /home/. /backups/home/
```

Warning ☠

Notice the relative path of *../* for *home.0/*. The path for the `--link-dest` argument should be relative to the target directory—in this case, */backups/home*. This has confused many people.

A simple example script

The following script can be run as many times as you want. The first time it runs, it creates the first "full backup" of */home* in */backups/home.inprogress* and moves that directory to */backups/home.0* upon completion. The next time through, `rsync` creates a hard-linked copy of */backups/home.0* in */backups/home.inprogress*, then uses that directory to synchronize to, updating any files that have changed, after

first unlinking them. This script then keeps three versions of the backups.

```
rsync -av --delete --link-dest=../home.0 /home/.
/backups/home.inprogress/
[ -d /backups/home.2 ] && rm -rf /backups/home.2
[ -d /backups/home.1 ] && mv /backups/home.1 /backups/home.2
[ -d /backups/home.0 ] && mv /backups/home.0 /backups/home.1
[ -d /backups/home.inprogress ] && mv /backups/home.inprogress
/backups/home.0
touch backups/home.0
```

This is a very basic script. If you're serious about implementing this idea, you have two options: either go to Mike Rubel's web page at http://www.mikerubel.org/computers/rsync_snapshots or look at the section on `rsnapshot` later in this chapter. It is a full implementation of this idea, complete with a user group that supports it.

Restoring from the Backup

Because backups created this way are just conventional Unix filesystems, there are as many options for restoring them as there are ways to copy files. If your backups are stored locally (as on a removable hard disk) or accessible over a network filesystem such as NFS, you can simply `cp` files from the backups to */home*. Or better yet, `rsync` them back:

```
$ rsync -av --delete /backups/home.0/. /home/
```

Be careful with that `--delete` flag when restoring: make sure you really mean it!

If the backups are stored remotely on a machine you can access by `ssh`, you can use `scp` or `rsync` rather than `ssh`. Other simple arrangements are also possible, such as placing the directories somewhere that is accessible to a web server.

Things to Consider

Here are a few other things to consider if you're going to use the `rsync` method for creating backups.

How large is each backup?

One drawback of the `rsync`/hard-link approach is that the sharing of unchanged files makes it deceptively hard to define the size of any one backup directory. A normally reasonable question such as, "Which backup directories should I erase to free 100 MB of disk space?" cannot be answered in a straightforward way.

The space freed by removing any one backup directory is the total disk usage of all files whose only hard links reside in that directory, plus overhead. You can obtain a list of such files using the `find` command, here applied to the backup directory */backups/home.1/*:

```
$ find /backups/home.1 -type f -links 1 -print
```

The following command prints their total disk usage:

```
$ du -hc 'find /backups/home.1 -type f -links 1 -print' | tail -n 1
```

Deleting more than one backup directory usually frees more than the sum of individual disk usages because it also erases any files that were shared exclusively among them.

> **Tip ✎**
>
> This command may report erroneous numbers if the source data had a lot of hard-linked files.

A brief word about mail formats

There are a number of popular mail storage formats in use today. The venerable *mbox* format holds all messages of a folder in one large flat file. The newer *maildir* format, popularized by Qmail, allows each message to be a small file. Other database mail stores are also in use.

Of these, *maildir*s are by far the most efficient for the `rsync`/hard-link technique because their structure leaves most files (older messages) unchanged. (This would be true of the original `rsync`/hard-link method and of `rsnapshot`, which is covered later in the chapter.) For `mbox` format mail spools, consider `rdiff-backup` instead.

Other useful rsync flags

If you have a slow network connection, you may wish to use `rsync`'s `--bwlimit` flag to keep the backup from saturating it. It allows you to specify a maximum bandwidth in kilobytes per second.

If you give `rsync` the `--numeric-ids` option, it ignores usernames, avoiding the need to create user accounts on the backup server.

Backing up databases or other large files that keep changing

The method described in this chapter is designed for many small files that don't change very often. When this assumption breaks down, such as when backing up large files that change regularly (such as databases, *mbox*-format mail spools, or UML COW files), this method is not disk-space efficient. For these situations, consider using `rdiff-backup`, covered later in this chapter.

Backing up Windows systems

While `rsync` works under *cygwin*, issues have been reported with timestamps, particularly when backing up FAT filesystems. Windows systems traditionally operate on local time, with its daylight savings and local quirks, as opposed to Unix's universal time. At least some file timestamps are made only with two-second resolution. Consider giving `rsync` a `--modify-window` of 2 on Windows.

Large filesystems and rsync's memory scaling

While it has improved in recent years, `rsync` uses a lot of memory when synchronizing large file trees. It may be necessary to break up your backup job into pieces and run them individually. If you take this route, it may be necessary to manually `--delete` pieces from the destination that have been deleted from the server.

Atomicity and partial transfers

`rsync` takes a finite period of time to operate, and when a source file changes while the backup is in progress, a partial transfer error (code 23) may be generated. Only the files that have changed may not be transferred; `rsync` completes the job as much as possible.

If you run backups only when your system is relatively static, such as in the middle of the night for an office environment, partial transfers may never be a problem for you. If they are a problem, consider making use of `rsync`'s `--partial` option.

You Put It Where?

A new VP came over and asked us to help him with a PC problem. When I start fixing the problem, I noticed that his PC was incredibly slow. His drive was 99% full and

really fragmented. I emptied the recycle bin, and defragged the drive. All problems disappeared.

That’s when the other shoe dropped.

He organized things by putting all his important files in the recycle bin. The Delete key was handy, and it put things there very quickly. (It was one of three icons on his desktop. Pretty handy, huh?) He claimed it was something that he was taught in the military.

Of course, our backup program didn’t back up the recycle bin.

Mark

rsnapshot

`rsnapshot` is an open-source backup and recovery tool based on Mike Rubel's original concept. It is designed to take a great idea and make it more accessible to users and more useful for larger environments. Under the hood, `rsnapshot` works in the same way as Mike Rubel's original scripts covered in the first part of the chapter. It uses hard links to conserve space and `rsync` to copy changes and break the hard links when necessary.

The major differences between `rsnapshot` and Mike Rubel's original script are that `rsnapshot`: [1]

- Is written in Perl because Perl skills are readily available, and Perl can more easily parse a configuration file.
- Supports multiple levels of backup (daily, weekly, and monthly) from one script.
- Backs up multiple directory trees from multiple sources, with some optional fine-tuning of what `rsync` parameters are used.
- Supports locking so that two backups don't run at once.
- Has `syslog` support.
- Supports pre- and post-backup scripts, which are useful for things like taking dumps of databases.

Platform Support

`rsnapshot` is most at home on a Unix-like platform, such as Linux, FreeBSD, and Solaris, but it can be used to back up data on non-Unix platforms. `rsnapshot` itself needs to run somewhere that `rsync` can deal with hard links, which means somewhere like Unix. For these purposes, Mac OS X counts as a Unix platform, although there are some issues with Mac

OS X's case-insensitive filesystem (HFS+) and resource forks. Thankfully, this is very rarely an issue. Resource forks are nowhere near as important in Mac OS X as they used to be in "classic" Mac OS. Please read the section on this topic in Chapter 3.

> **Tip ✎**
>
> Classic Mac OS is not supported at all. While it may be possible to get an `rsync` server running on it and for `rsnapshot` to back it up, no changes will be made to `rsnapshot` to support this.

`rsnapshot` can back up Windows systems, but running the `rsnapshot` server on Windows is not supported because of the lack of hard-link support. Also, because permissions are so different in the Windows world compared with the Unix world, they will almost certainly not be backed up properly. Timestamps on files may not be backed up properly either, because Windows seems to store timestamps in the local timezone whereas Unix uses GMT. However, with these caveats, `rsnapshot` is being used successfully by many people to back up their Windows machines. To back up permissions, you can run a script before the backup to dump information about file permissions and users to a file that is then included in the snapshot. When you recover a file from backup, you then restore the correct permissions to files either by hand or with a script, referring to that file. Obviously, you should test this procedure before relying on it, to make sure you are recording enough data and that your script is correctly reading the file!

Finally, some obscure features of Unix filesystems aren't backed up because `rsync` doesn't support them: ACLs and filesystem extended attributes. These are rarely used, and you can use the same workaround described for Windows. No scripts of this nature are supplied with `rsnapshot` because

there is so much variety from one platform to another (which is the same reason that `rsync` itself doesn't support them).

> **Tip ✎**
>
> `rdiff-backup` often supports backing up ACLs.

When Not to Use rsnapshot

With even the slightest change to a file, `rsnapshot` puts a new copy of the file into your backup. This works well if you add a file, remove a file, or make large changes to a file. However, it does fall down if you either make small changes to a big file or are constantly changing a file. In those cases, you lose all the benefits of hard links. (`rdiff-backup` is covered later in the chapter.) Files where these sorts of change are common are logfiles and mboxes. Normally, these make up only a small proportion of the data on a machine, and so the amount of wasted space is actually quite small compared to the size of a backup. However, in the specific cases of mail servers and syslog servers, pretty much the only changes to files are of this sort. In those cases, we recommend using `rdiff-backup` instead.

Setting Up rsnapshot

`rsnapshot` packages are available for several Linux distributions and can be easily installed using the native package manager. If your distribution or OS doesn't have such a package available or if you want to be up to date with the most recent version, you can download a gzipped `tar` file from:

http://www.rsnapshot.org/downloads.html

To install that version, first uncompress the file:

```
$ gzip -d rsnapshot-version.tar.gz | tar xvf -
```

Then change to the directory this command created:

```
$ cd rsnapshot-version
```

There is a file of instructions called *INSTALL*, but, in summary, for a fresh install you would normally need to issue these commands:

```
$ ./configure
$ su
 (you will be asked for your password)
# make install
#cp /etc/rsnapshot.conf.default /etc/rsnapshot.conf
```

You then edit the file */etc/rsnapshot.conf* to tell it where to put your backups, what to back up, and so on. There are copious instructions in the example config. Most users need to change only these settings:

`backup`

: What to back up

`snapshot_root`

: Where to put your backups

`interval`

: When to back up

`include` and `exclude`

: Which files `rsync` pays attention to

You can check your new configuration by typing:

```
$ rsnapshot configtest
```

If it all looks good, create `cron` jobs to run `rsnapshot` automatically. You can put them in root's `cron` file like this:

```
MAILTO=myemail@example.com # so that I get emailed any errors
00 00 * * *    /usr/local/bin/rsnapshot daily    # every day at
midnight
00 22 * * 6    /usr/local/bin/rsnapshot weekly   # every Saturday
```

```
at 22:00
00 20 1 * *    /usr/local/bin/rsnapshot monthly  # the first of the
month at 20:00
```

If you want to back up data from other machines, use `ssh` (which must be installed on both ends of the connection) and keys. There's a good description of how to do this securely here:

http://troy.jdmz.net/rsnapshot/

There's also a web frontend to the CVS repository here:

http://rsnapshot.cvs.sourceforge.net/rsnapshot/rsnapshot/

The rsnapshot Community

Open-source software lives or dies by its support. Thankfully, `rsnapshot` is well supported by an active mailing list that users are encouraged to subscribe to. It is used for reporting problems, discussing their fixes, announcing new releases, and so on. You can join the list or search the archives here:

https://lists.sourceforge.net/lists/listinfo/rsnapshot-discuss

Many people post patches and add-ons to the list, some of which are committed to CVS by the small number of developers. If you've got good ideas and have posted good patches to the list, getting permission to commit to CVS is easy. Of course, everyone can look at and download the CVS repository anonymously.

`rsnapshot` is stable software. There are no plans to make major changes in how it works. The sort of changes you can expect to see are:

- More add-on scripts for reporting
- More database support scripts
- More configuration options
- Occasional bug fixes

[1] Some of these differences no longer apply because Mike has continued to update his script. Specifically, the new version of Mike Rubel's script supports multiple trees and locking, and it also has a `syslog` support wrapper.

rdiff-backup

`rdiff-backup` is a program written in Python and C that uses the same rolling-checksum algorithm that `rsync` does. Although `rdiff-backup` and `rsync` are similar and use the same algorithm, they do not share any code and must be installed separately.

When backing up, both `rsnapshot` and `rdiff-backup` create a mirror of the source directory. For both, the current backup is just a copy of the source, ready to be copied and verified like an ordinary directory. And both can be used over `ssh` in either push or pull mode. The most important conceptual differences between `rsync` snapshots and `rdiff-backup` are how they store older backups and how they store file metadata.

An `rsync`-snapshot system basically stores older backups as complete copies of the source. As mentioned earlier in the chapter, by being clever with hard links, these copies do not take long to create and usually do not take up nearly as much disk space as unlinked copies. However, every distinct version of every file in the backup is stored as a separate copy of that file. For instance, if you add one line to a file or change a file's permissions, that file is stored twice in the backup archive in its entirety. This can be troublesome especially with logfiles, which grow slightly quite often.

On the other hand, `rdiff-backup` does not keep complete copies of older files in the backup archive. Instead, it stores only the compressed differences between current files and their older versions, called *diffs* or *deltas*. For logfiles, `rdiff-backup` would not keep a separate copy of the older and slightly shorter log. Instead, it would save to the archive a delta file that contains the information "the older version is

the current version but without the last few lines." These deltas are often much smaller than an entire copy of the older file. When a file has changed completely, the delta is about the same size as the older version (but is then compressed).

When an `rdiff-backup` archive has multiple versions of a file, the program stores a series of deltas. Each one contains instructions on how to construct an earlier version of a file from a later one. When restoring, `rdiff-backup` starts with the current version and applies deltas in reverse order.

Besides storing older versions as deltas instead of copies, `rdiff-backup` also stores the (compressed) *metadata* of all files in the backup archive. Metadata is data associated with a file that describes the file's real data. Some examples of file metadata are ownership, permissions, modification time, and file length. This metadata does not take up much space because metadata is generally very compressible. Newer versions go further and store only deltas of the metadata, for even more space efficiency.

At the cost of some disk space, storing metadata separately has several uses: first, data loss is avoided even if the destination filesystem does not support all the features of the source filesystem. For instance, ownership can be preserved even without root access, and Linux filesystems with symbolic links, device files, and ACLs can be backed up to a Windows filesystem. You don't have to examine the details of each filesystem to know that the backup will work. Second, with metadata stored separately, `rdiff-backup` is less disk-intensive on the backup server. When backing up, `rdiff-backup` does not need to traverse the mirror's directory structure to determine which files have changed. Third, metadata such as SHA-1 checksums can be used to verify the integrity of backups.

Advantages

Here are some advantages of using `rdiff-backup` instead of an `rsync` script or `rsnapshot`:

Backup size

> Because `rdiff-backup` does not store complete copies of older files but only the compressed differences between older and current files, backups generally consume less disk space.

Easier-to-use

> Unlike `rsync`, `rdiff-backup` was written originally for backups. It has sensible defaults (so no need for the `-av --delete -e ssh` options) and fewer quirks (for instance, there is no distinction between *<destination>*, *<destination>/*, and *<destination>/.*).

Preserves all information

> With `rsync`, all information is stored in the filesystem itself. If you log in to your backup repository as a nonroot user (generally a good idea), the `rsync` method forgets who owns all your files! `rdiff-backup` keeps a copy of all metadata in a separate file, so no information is lost, even if you aren't root or if you back up to a different kind of filesystem.

Handy backup features

> `rdiff-backup` has several miscellaneous handy features. For example, it keeps detailed logs on what is changing and has commands to process those logs so that you know which files are using up your space and time. Also, newer versions keep SHA-1 checksums of all files so you can verify the integrity of backups. Some `rsync` scripts have similar features—check their documentation.

Disadvantages

Let's be honest. `rdiff-backup` has some disadvantages, too:

Speed

> `rdiff-backup` consumes more CPU than `rsync` and is therefore slower than most `rsync` scripts. This difference is often not noticeable when the bottleneck is the network or a disk drive but can be significant for local backups.

Transparency

> With `rsync` scripts, all past backups appear as copies and are thus easy to verify, restore, and delete. With `rdiff-backup`, only the current backup appears as a true copy. (Earlier backups are stored as compressed deltas.)

Requirements

> `rdiff-backup` is written in Python and requires the `librsync` library. Unless you use a distribution that includes `rdiff-backup` (most of them include it), installation could entail downloading and installing other files.

Quick Start

Here's a basic, but complete, example of how to use `rdiff-backup` to back up and restore a directory. Suppose the directory to be backed up is called <*source*>, and we want our archive directory to be called <*destination*>:

```
$ rdiff-backup source destination
```

This command backs up the *<source>* directory into *<destination>*. If you look into *<destination>*, you'll see that it is just like *<source>* but contains a directory called *<destination>/rdiff-backup-data* where the metadata and deltas are stored. The *rdiff-backup-data* directory is laid out in a fairly straightforward way—all information is either in (possibly gzipped) text files or in deltas readable with the `rdiff` utility—but we don't have the space to go into the data format here.

The first time you run this command, it creates the *<destination>* and *<destination>/rdiff-backup-data* directories. On subsequent runs, it sees that *<destination>* exists and makes an incremental backup instead. For daily backup usage, no special switches are necessary.

Suppose you accidentally delete the file *<source>/foobar* and want to restore it from backups. Both of these commands do that:

```
$ cp -a destination/foobar source
$ rdiff-backup -r now destination/foobarsource
```

The first command works because *<destination>/foobar* is a mirror of *<source>/foobar*, so you can use `cp` or any other utility to restore. The second command contains the `-r` switch, which tells `rdiff-backup` to enter restore mode, and restore the specified file at the given time. In the example, `now` is specified, meaning restore the most recent version of the file. `rdiff-backup` accepts a large variety of time formats.

Now suppose you realize you deleted the important file *<source>/foobar* a week ago and want to restore. You can't use `cp` to restore because the file is no longer present in *<destination>* in its original form (in this case it's gzipped in the *<destination>/rdiff-backup-data* directory). However the `-r` syntax still works, except you tell it `7D` for seven days:

```
$ rdiff-backup -r 7D destination/foobarsource
```

Finally, suppose that the <*destination*> directory is getting too big, and you need to delete older backups to save disk space. This command deletes backup information more than one year old:

```
$ rdiff-backup –-remove-older-than 1Ydestination
```

Just like `rsync`, `rdiff-backup` allows the source or destination directory (or both) to be on a remote computer. For example, to back up the local directory <*source*> to the <*destination*> directory on the computer *host.net*, use the command:

```
$ rdiff-backup sourceuser@host.net::destination
```

This works as long as `rdiff-backup` is installed on both computers, and *host.net* can receive `ssh` connections. The earlier commands also work if *user@host.net::*<*destination*> is substituted for <*destination*>.

Windows, Mac OS X, and the Future

Although `rdiff-backup` was originally developed under Linux for Unix-style systems, newer versions have features that are useful to Windows and Mac users. For instance, `rdiff-backup` can back up case-sensitive filesystems and files whose names contain colons (:) to Windows filesystems. Also, `rdiff-backup` supports Mac resource forks and Finder information, and is easy to install on Mac OS X because it is included in the Fink distribution. Unfortunately, `rdiff-backup` is a bit trickier to install natively under Windows; currently, `cygwin` is probably the easiest way.

Future development of `rdiff-backup` may consist mostly of making sure that the newer features like full Mac OS X support are as stable as the core Unix support, and adding

support for new filesystem features as they emerge. For more information on `rdiff-backup`, including full documentation and a pointer to the mailing list, see the `rdiff-backup` project home page at http://rdiff-backup.nongnu.org/.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Part 3. Commercial Backup

Part III consists of two chapters:

Chapter 8, Commercial Backup Utilities

Explains the features of commercial products so you can assess their usefulness.

Chapter 9, Backup Hardware

Explains the many different types of backup hardware available today and provides criteria to help you decide which type of backup drive is right for you.

Chapter 8. Commercial Backup Utilities

The main purpose of this book is to help those with smaller budgets figure out how to back up their systems without spending a lot of money. Toward that end, we have explained the basic backup tools that are available in Unix, Windows, and Macintosh systems. We have also covered four different open-source backup and recovery tools that can meet the data protection needs of many companies.

For those who cannot meet their backup and recovery requirements with open-source utilities, there is an entire industry with commercial backup utilities available to you. Remember that just being a commercial product does not make a given product better than an open-source product. Some commercial products offer less functionality than that which is provided by the open-source tools covered in this book. Others, however, offer significantly more functionality and may be used to solve problems unsolvable with open-source tools.

By explaining the various features that commercial backup products have, we hope to help you decide:

When it might be appropriate to purchase a commercial backup product

Which commercial product might be right for you

Choosing a commercial backup product is hard work. There are more than 50 products with hundreds of features that change every day. Combine the complexity of the subject matter with the fact that every company’s data protection needs are different, and the result is that if two administrators of equal skill from two different companies perform an equally exhaustive search for a backup product, they will arrive at different results because of differences in

their companies' specific needs. A product that might not be a good choice for any other company might be the *perfect* choice for yours, simply because it does something no other product does—something that your company needs it to do.

Although there are a few products that come close, there is no single product that meets everyone's needs. That means that neither this chapter nor this book will "pick product X" at any time. This is for a lot of reasons, not the least of which is *change*. The data protection market changes every day. Users' data protection needs change, and what different companies do to meet those needs changes. Add to that the ever-present influence of competition. It's happened more than once that a lesser-known backup product comes out with a new version that significantly changes its standing in the market. The constantly changing nature of the data protection market means that any recommendation you read here could be wrong by the time the book hits the shelves, so we dicuss features only.[1]

Neither will this book attempt to give a summary description of available backup products. There are two reasons for this. The first reason is that the information here may be out of date by the time the book is printed. The second reason is bias. I would be lying if I said that I knew all these products equally. I know certain products better than others, and those products would receive a more accurate description.

What to Look For

When looking at a potential backup product, there are many questions that you should ask yourself:

- Does the product fully support my platforms?
- Does the product back up raw partitions?

- Does the product back up very large filesystems and large files?
- How will the product help me meet extremely aggressive requirements?
- Does the product back up many clients to one drive simultaneously?
- Can the product automatically create multiple simultaneous backups from one client?
- Does the product handle data requiring special treatment?
- Does the product have storage management features?
- Does the product reduce network traffic?
- Does the product support a standard or unique backup format?
- How easy is the product to administer?
- How secure is the product?
- How easily does the product perform recoveries?
- How well does the product protect the backup catalog (e.g., database, index)?
- How robust is the product?
- How automated is the product?
- Can the product verify its volumes?
- What does the product cost?
- What vendor is selling the product?

These issues are discussed in the following sections.

---

[1] Just to illustrate this point, a few years ago I really liked a particular backup product. It was a very good product that had features that other products still don’t have. The company was bought out, and the product no longer exists!

Full Support of Your Platforms

One of the easiest ways to narrow the list of backup software vendors from which to choose is to find out who supports the platforms you are running. There is no reason that a vendor should have to answer an RFI with hundreds of questions if it doesn't support most of your platforms. By the same token, there is no reason you should have to read the answers to hundreds of responses from more than 50 vendors. Before getting into the nitty-gritty features, simply find out who supports all or most of the platforms that need to be backed up.

The key word in the previous paragraph is "most." Most shops are becoming increasingly heterogeneous, with a number of Unix variants, Intel-based operating systems, and any number of database-like products. In such a shop, there are always a few boxes that run a "different" version of Unix, an older version of Windows, some other operating system, or even a lesser-known database product that most products don't support. During the first cut, include products that back up *most* of the operating systems that you are running. Restricting the search to only those that handle every platform might exclude some very good products or force the selection of the wrong product. Many vendors will consider porting their products to a new operating system or database if a potential customer asks them to do so, although this may come at a cost. (In the final analysis, however, a product that has been supporting a particular platform for a while *usually* supports it better.)

There are many different operating systems out there. There are Unix variants, Mac OS, various Windows flavors, NetWare, and mainframe operating systems. Most of the big products handle almost all of these. There was a time when you had to purchase one product for the Novell

servers, one for Windows, one for Mac OS, one for the MVS mainframes, and yet another for the Unix servers. Today, some products handle all of those from one console. However, the rush by so many large backup products to support *everything* has left a few holes. There are a number of examples of this. Things that are typically left out in Unix are block special files, character special files, and named pipes. Microsoft's System State and Active Directory also are occasionally left out, Mac OS resource forks are forgotten, and Novell's NDS is left out as well. Make sure each product fully supports the platforms it claims to support.

Should You Back Up Special Files?

If a backup product did not back up the Windows Registry, it would not be considered certified by Microsoft. However, some prominent Unix backup packages don't back up special files and named pipes. (These are special types of files in Unix and Linux.) If the operating system is lost, their typical answer is to "reinstall the operating system and our software, and then start the restore." This takes too much time and involves writing a lot of information twice. If the backup system backed up the special files, there is a much quicker way to recover from the loss of the operating system. See Part IV of this book for more information about such recoveries.

What if a system was able to boot, but the */dev* directory was all messed up? Being able to restore those files can reduce the number of hours of downtime. Does the backup product really need to back up the device files? Yes it does!

Back It Up — All of It!

A production system had a bad disk and lost the programs that ran the application, including the database. The normal backup backed up only the data, not the application software. My coworkers could not find the original software. Fortunately, I had a complete tape copy in my desk from testing before production, and we were able to use it to restore the software. (Eventually, the original software also was found, copied, properly stored, and cataloged.)

Norm Eisenberg

Backup of Raw Partitions

Many environments use their commercial backup product to back up raw partitions. A *partition* is a section of disk that may or may not contain a filesystem. Typically (although not always), when you refer to a *raw* partition, you are referring to a section of disk that does *not* contain a filesystem. This disk may contain data for a database product, such as Oracle, Informix, or Sybase. It also may be the first part of the root partition of the operating system disk that contains the boot block. Since most backup products are designed to back up files that reside on a filesystem, they may not be able to back up a raw partition.

The ability to back up raw partitions could help when backing up relatively small databases that reside on raw partitions. To back up most databases with a product that supports raw partitions, simply shut down the database and tell the backup software what raw partitions to back up. In order to do this, the backup software needs to be able to back up these raw partitions.

The second reason to consider the use of this feature is to back up the root partition of an operating system disk. It’s another way to recover the root disk without reinstalling the operating system. There are two essential parts to the operating system disk. The first is the operating system itself, which resides on one or more filesystems on that disk. The second is the boot block (in Unix variants) or Master Boot Record (in Intel systems). This tells the system’s firmware where to go to find the operating system kernel. This block of data normally can reside on the first slice, or root partition, of the operating system disk. In modern Linux systems, it resides in a special partition called */boot*. It resides *outside* the “normal” filesystem and thus is not backed up by normal procedures. If the backup

product is able to back up the raw partition on which it resides, it’s possible to recover it without reinstalling the operating system. (This is covered in detail in Part IV of this book.)

Many of the popular backup packages now work with raw partitions. There is one drawback to backing up raw partitions, though. A raw partition is seen as one big file. That means that every time it is backed up, the entire partition is backed up. With a 100 MB root partition of an operating system, this is not a problem. If it’s a multiterabyte raw device, it can fill up quite a lot of backup media very fast. Some products can intelligently read a raw partition and perform an incremental backup of its contents. It is a rare option as of the writing of this book, but it is worth investigation.

Backup of Very Large Filesystems and Files

Large filesystems and files caught many backup products by surprise. For many years, 4 GB filesystems with a maximum of 2 GB files ruled the land. This is because none of the operating systems allowed anything larger. Then came 32-bit and 64-bit operating systems and the multiterabyte filesystem. Not long after that, some vendors were announcing the ability to create multiterabyte files. This caused a major problem with some backup vendors because many design decisions were made assuming there was a 4 GB limit.

> **Tip ✎**
>
> It’s inexcusable that a product would still have this problem at this point.

There are a few things to consider when investigating whether a given product can handle large files and filesystems. Does the vendor have any hardcoded limits that say a file can’t be any bigger than N bytes? Do they have problems if a filesystem (or file) is bigger than a volume? Do they have any automated way to create multiple simultaneous backups of a single filesystem, without requiring you to manually divide that filesystem into many pieces?

Switching Backup Products

You should never change your backup product just because you’re having problems with it. First, the problems with any given backup system are almost always misconfiguration, misunderstanding, lack of defined processes, not enough hardware, or too much

hardware. It's almost *never* the product itself. Second, switching backup products affects all three important business factors: cost, risk, and service levels. There's the cost of acquiring the product, legacy restores, new product training, and implementation services. Then there's the risk of data loss while you are learning the new product. Thirdly, there will be an apparent drop in service levels as you try to figure out how the new system works.

Therefore, if you're having problems with your backup system, switching products should be the last thing on the list. Before you do that, hire a specialist in your backup product to make sure that it's performing optimally and that you've taken advantage of any useful extra features that might help solve your problem. They'll probably solve your problem, it will cost a fraction of what a new backup product will cost, and your knowledge of your current product will increase. Therefore, there's only one reason that you should be considering changing your backup product: you have requirements that your current backup product cannot meet.

Aggressive Requirements

As mentioned in the "Switching Backup Products" sidebar, there's only one reason that you should consider changing your backup product: you have requirements that your current backup product cannot meet. These requirements include your recovery time objective, recovery point objective, consistency, and backup window groups.

Your *recovery time objective*, or RTO, is how quickly you want the system to be recovered. RTOs can range from zero seconds to many days, or even weeks. Each piece of information serves a business function, so the question is how long you can live without that function. If the answer is that you can't live without it for one second, you have an RTO of zero seconds. If the answer is that you can live without it for two weeks, you have an RTO of two weeks.

The *recovery point objective*, or RPO, is determined by how much data you can afford to lose. If you can lose three days worth of a given set of data, that set of data has an RPO of three days. If it's real-time customer orders, however, you may decide you can't afford to lose any of them; you have an RTO of zero for that application.

There can also be an RPO for a group of machines. If you have several systems that are related to each other, you may need to recover them to the same point in time. They are referred to as a *consistency group*. To meet such a requirement, you have to back up all related systems at exactly the same time, or you have to give each system a very small RPO. Having an RPO for a group of machines basically makes the RPO for each machine in that group the same as the lowest RPO of any machine in that group.

Once you've determined an RTO and RPO for each system and disaster type, you need to agree on when you can back

up a system, how long you can take to back it up, and how much you are allowed to impact the production system while it is being backed up. These values are collectively and generally referred to as the *backup windo w*.

Once you've determined your requirements, you may find that they are too aggressive. We'll consider a requirement *aggressive* if a traditional LAN-based recovery method can't meet it. This typically means that the bandwidth is too small, the amount of data to move is too large, or the amount of time you're given isn't reasonable. This tends to happen in one of three scenarios:

Remote office backup

: Remote offices have long been the elephant in the room at many data protection planning sessions. They're typically handled with remote tape drives or tape libraries that aren't being managed by skilled personnel, or they're not being managed at all. I recently met with an oil-and-gas company whose remote offices included off-shore drilling rigs. Imagine the fun they have getting a vaulting company to stop by!

  However, an increasing number of people want to fix this problem by backing up their remote office across the network. How do you back up a remote office with hundreds of gigabytes of data if it's on the other side of a WAN connection? A typical backup and recovery system would not be able to meet any reasonable RTO, RPO, or backup window requirements.

Very large applications

: I've recently seen a 150 TB Oracle database. Try backing that thing up! While you may not have a 150 TB application, it's highly possible that you have an application that's too large to back up and recover within an acceptable time. As of this writing, that size

seems to be individual servers that have gone beyond several terabytes. Anything a few terabytes of smaller can be backed up and recovered within a few hours, which is within range of most RTOs and RPOs. However, what do you do if you've got a 10 TB application and a one-hour RTO?

Very critical applications

Some applications are so critical to the business that their owners simply won't accept any downtime or loss of data. How do you meet a one-minute RTO or a zero-second RPO, regardless of how large your application is? Applications that fall into this category are the hardest applications to design for and require very advanced data protection systems.

The following technologies can help you meet aggressive requirements, and they are listed in order of their ability to meet such requirements. The farther down the list they appear, the more aggressive requirements they can meet.

Test Your Restores

At a large insurance company, a decision had been made to back up our NetApp Filer by backing up the shares rather than purchasing an NDMP key and dedicated tape drive. We used the utility from the Windows resource pack to mount the share at boot time. For an unknown reason, the share did not get mounted, and because we specified `soft` in the mount configuration file, there was no notice or error message given.

Since the share was never mounted, there was no notice given during backups. However, if the owner of the share logged on to the system, the share would be mounted at that time and dismounted when he signed off. This went on for several months before the owner of the share

> corrupted the data and asked for a restore from the previous night. It was then that we found out there were no backups for three months and that the share he was using had expired.
>
> *Harry Tirrell*

LAN-Free Backup

The first advanced backup and recovery technique is to send the backup data across a SAN instead of the LAN—thus earning the term *LAN-free*. Since LAN-free backups are faster than LAN-based backups, they can be used to help meet more aggressive RTOs and RPOs than you could meet with LAN-free backups. If you've got a large server that's having trouble meeting its RTO or RPO, you may consider making it a LAN-free backup client.

LAN-free backups require the backup device to be connected via a storage area network, or SAN, running either Fibre Channel or iSCSI. A Fibre Channel SAN would use Fibre Channel HBAs and a Fibre Channel switch. An iSCSI SAN would use regular HBAs with iSCSI drivers (or perhaps iSCSI HBAs), and can route their traffic across any IP network—although it's a good idea to separate this traffic from regular IP traffic.

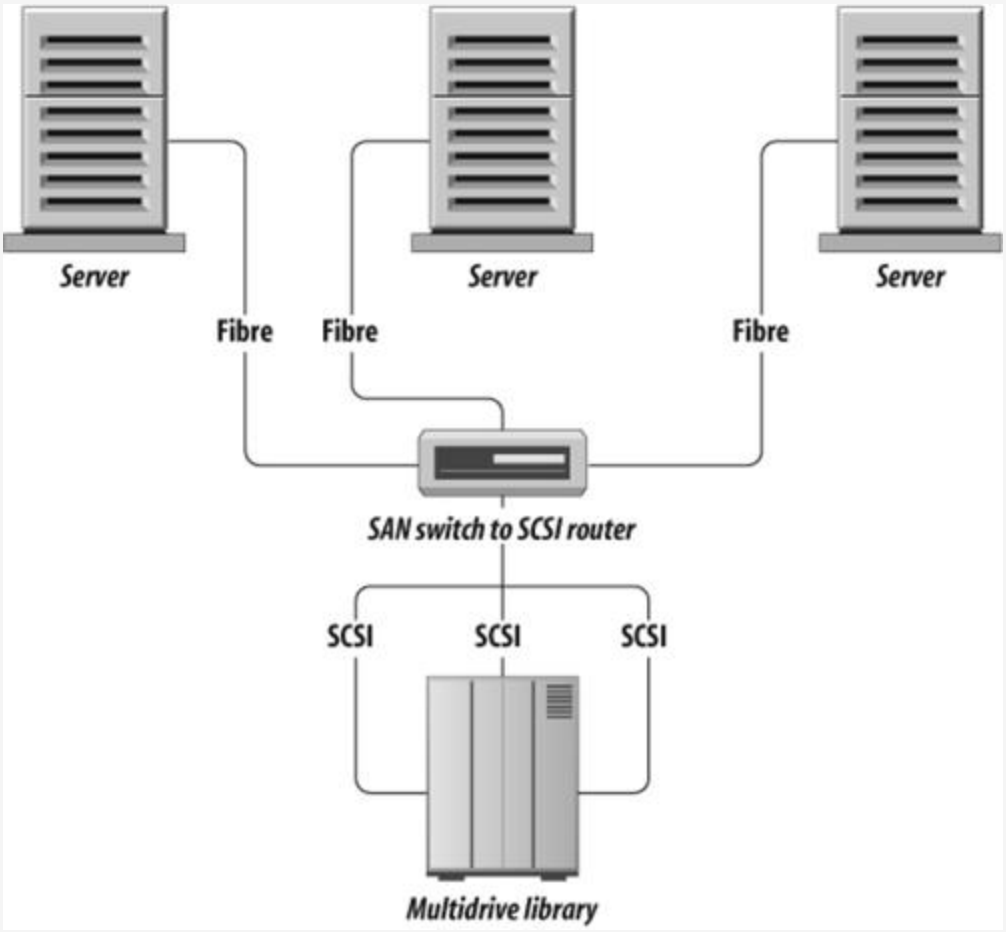


Figure 8-1. A basic storage area network (SAN)

As illustrated in Figure 8-1, backup servers connected to a SAN have *virtual* physical access to all peripherals in the SAN. Once the peripherals are attached and the SAN is configured, all SCSI/Fibre Channel/iSCSI traffic is routed through the SAN, and each server “thinks” that the library is locally attached to it. This allows the server to take advantage of the recent advancements in backup technology that allow backups to be sent across the SAN. They are much faster (and much easier on the CPU) than LAN-based backups.

Tip ✎

When people first see a SAN drawing, they don't see much difference between it and a LAN, and the line between the two gets more blurred every day. Historically, a SAN used Fibre Channel, and a LAN used Ethernet and IP. Now with iSCSI, a SAN can also use IP as a transport mechanism. Just remember that a SAN is talking SCSI. That SCSI may be running on top of Fibre Channel or IP—but it's still SCSI.

Commercial backup applications can dynamically configure which servers have access to each peripheral. For example, when it is time for a particular server's full backup, the backup application can configure the router in such a way that it has access to every available backup drive. Of course, it can do this for a critical restore as well.

Server-Free (or Serverless) Backup

LAN-free backups do make backups go much faster, and they do require less CPU than LAN-based backups, but they also still require some CPU. If you have a very large server, even that reduced CPU load may be more than the server can handle. It may impact the application too much, or it may require more CPU than you have available. What if you could back up the application without actually sending the data through the server that is using the database? If you're willing to accept a very different type of backup design, this is now possible. Consider the drawing in Figure 8-2.

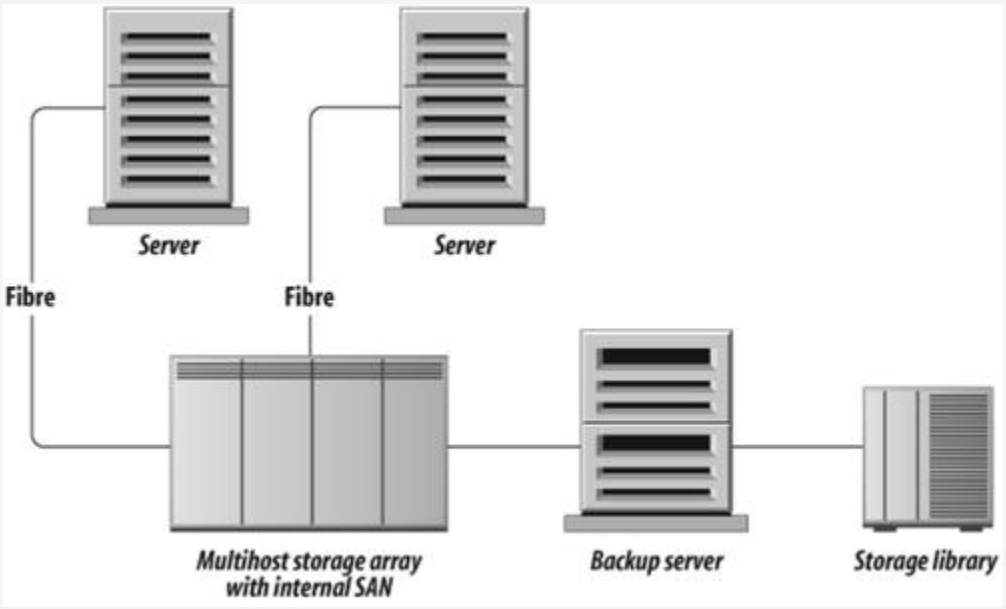


Figure 8-2. Serverless backup

At the top of Figure 8-2, there are two database servers that are connected to a large, multihost-attachable disk array. The backup server also is attached to this array. The databases actually are sitting on top of mirrored partitions inside the large disk array. To accomplish a *serverless backup*, the backup server tells the database server that it needs to do a backup of the database. The database server splits off one of the mirrors or makes a virtual snapshot of its volumes, then tells the backup server that it can back it up. The backup server then backs up the data via a path that doesn't include the original database server—hence the term *serverless*.[2] (Of course, the backup server is still involved. It's just the original server that no longer has to move the data.) The entire application can then be backed up without transferring the data through the client that is using it. The data may take one of two paths:

- It can back up the data via another dedicated server that can access the split-off mirror or snapshot.
- It can back up the data via a SAN router than accepts the SCSI X-COPY command. This moves the data

directly from the disk device to tape without going through *any* server.

If the data being backed up is a split mirror (instead of a virtual snapshot), it also offers another advantage that traditional backup methods cannot. This second mirror can be left disconnected until it is time to back it up again. At that point, it can be quickly resynced to the other side of the mirror. Leaving it disconnected like this gives you an instantly available backup of the entire database. If something were to happen to the production database, you could run a few commands and remount the database using the mirror. All you would need to do is to replay your transaction logs since the mirror was split off. Without this standby mirror, you would have to restore the database before you could replay your transaction logs. This technology could be used to meet very aggressive RTOs and RPOs.

There are a few disadvantages to this method, starting with the fact that it is extremely complex. If everything goes right, you're fine. If something goes wrong, you've got logs on the backup server, media server, client, storage array, and SAN router. It can take a really long time to figure out why it's not working. The second disadvantage is that most serverless backup products don't offer serverless restores. Make sure to look into that when investigating this option. Finally, this is also a very expensive option.

Tip ✎

Unix Backup & Recovery spoke more highly of serverless backup. A lot of things have changed since then, starting with the three technologies that will be covered next. Many people now consider them the preferred methods for meeting very aggressive RTOs and RPOs.

De-Duplication Backup Systems

The developers of *de-duplication backup systems* asked themselves a few questions. If only a few bytes change in a file, why are we backing up the entire file? If the same file resides in two places on the same system, why do we back it up twice? Why don't we just store a reference to the second file? Some even asked why we're backing up the same file across multiple systems. Doesn't that waste server and network resources?

The answers to all of these questions, of course, rest in the limitations of traditional backup systems. If we don't back up the file every time we see it, we're going to need to load a lot more tapes when we have to restore. In addition, if we only back up the changed bytes in a file, we might need multiple tapes just to restore a single file.

However, if you back up any given file only once, and back up the changed bytes only when a file changes, it is actually possible to meet more demanding backup window requirements. The tape issues mentioned here are mitigated by backing up to disk. Tape-based copies of the disk-based backups can be created at any time, depending on the requirements of the customer. Some de-duplication products can also meet aggressive RTO requirements by

restoring only the blocks that have changed since the file was last backed up. The RPO abilities of these products are based on how often you back up, but it is common to use such products to back up hourly, allowing you to meet a one-hour RPO. The same is true for your consistency group requirements.

> **Tip ✎**
>
> De-duplication backup systems use techniques similar to those used by disk targets with de-duplication features, which are discussed in Chapter 9. Where those systems de-duplicate at the target level, a complete de-duplication backup system eliminates the redundancy at the client level, reducing the amount of data that has to be sent from a remote office or laptop.

The biggest advantage to de-duplication products is that, from the user adoption perspective, they're the closest to what users already know. Their interfaces are similar, and they often have database agents similar to those found in traditional backup software. They're simply able to back up faster and more often, and they use much less bandwidth.

Snapshots

Another alternate backup method is a snapshot. The most common type of snapshot is a *virtual* copy of a device or filesystem that relies on the original volume to actually present the data to you.[3] This reliance on the original volume is why snapshots must be backed up to provide recovery from physical failures. Snapshot functionality may be found in a number of places, including advanced filesystems and volume managers, enterprise storage arrays, NAS filers, and backup software.

Snapshots can help meet aggressive backup requirements. For example, some snapshots can meet an RTO of a few seconds by simply changing a pointer. You also can create several snapshots per day, allowing for an aggressive RPO. Since snapshots can be created in seconds, you can meet aggressive backup window requirements as well. You can create a stable, virtual backup of a multiterabyte database in seconds, reducing the impact on the application to potentially nothing. Then you've got hours to perform a backup of that snapshot. The next section discusses how replication is a great way to do that. Finally, creating synchronized snapshots on multiple systems is also relatively easy, so you can meet aggressive synchronicity requirements as well.

One interesting development in the snapshot world is the development of APIs that allow other vendors to interface with snapshots. NDMP and Microsoft's VSS are examples. NDMP allows for backup vendors to schedule the creation of a snapshot, as well as catalog and restore from its contents. Restores are performed using the same interface that you would use for "normal" backups, but they are actually performed by the filer using snapshot technology. VSS allows storage vendors with snapshot capabilities to have the files in those snapshots listed in and restored from the Previous Versions tab in Windows Server 2003 and later. Hopefully, this functionality will be added to workstation versions of Windows, and more NAS vendors will support it as well.

Another interesting development with snapshots is the creation of database agents that work with snapshots. The database agent communicates with the database so that it believes it's being backed up when all that's really happening is the creation of a snapshot. Recoveries are also sometimes integrated, allowing for incredibly fast recoveries that are controlled by the database application.

Replication

Replication is the practice of continually copying from a source system to a target system all files or blocks that have changed on the source system. Replication used to be what companies implemented after everything was completely backed up and redundant, which meant that very few people used replication. However, many people are now using replication as their first line of defense for providing both backup and disaster recovery.

Replication by itself is not a good backup strategy; it copies everything, including bad things, such as viruses and file deletions. Therefore, a replication-based backup system must be able to provide history by either occasionally backing up the replicated destination or through the use of snapshots. It's usually preferable to make a snapshot on the source and replicate that snapshot to the destination. That way, you can prepare database applications for backup, take a snapshot, then have that snapshot replicated.

Near-Continuous Data Protection Systems

Replication, when coupled with snapshots, is called near-continuous data protection (or near-CDP). Since you can take snapshots hourly (or even more often in some systems), and the replication is occurring continuously, snapshots and replication are closer to CDP than traditional backup, hence the term *near-CDP*. Some near-CDP products replicate first, then take snapshots on the target system. Others take snapshots on the source system and replicate those snapshots to the target system.

One advantage of near-CDP systems is that snapshots take just seconds to create, and replication is a very easy way to

get the data to another device. You can also cascade replication to provide multiple copies, such as an on- and off-site copy, without touching a tape. If you then want to provide a tape copy of the replicated snapshot, you simply back up one of the destination devices.

The biggest disadvantage when compared to true CDP products is that when you cause logical corruption on the source system, such as deletion or corruption of a file, that corruption immediately overwrites the current backup, and you have to recover the file as of the most recent snapshot. A true CDP product would be able to recover the file to just before you fat-fingered it. Another disadvantage to near-CDP systems is that they often require you to change your primary storage system to support them because they are usually storage-array-based.

Continuous Data Protection Systems

A true continuous data protection (CDP) system is fundamentally an asynchronous replication-based backup system that doesn’t overwrite the target with the most recent data. The software is continuously running, and every time a file changes, the new bytes in that file are sent to the backup server within seconds or minutes. Unlike replication, however, a continuous data protection system stores the changes in a log instead of overwriting the target system with the most recent blocks; it is therefore able to roll back any changes at any time.

Different CDP products transfer data to the backup server in different ways. Some transfer changed blocks immediately while others batch up changed blocks and send them every few minutes. They also differ in how they do recoveries. Some products do quick restores by restoring only the blocks that have changed since the point

in time you are recovering from; others recover in a more traditional manner, recovering the entire file or filesystem that you asked to be recovered. Obviously, the first method allows you to meet much more aggressive RTOs and RPOs than the second method.

Either It's Continuous or It's Not!

Some vendors are referring to their near-CDP products as CDP. They're doing this to ride the market momentum that CDP has built. Some even defend that they're actually continuous because they are continually replicating. I heard one vendor say, "We're continuously copying all the data. We're just not keeping it all!" It reminds me of the Seinfeld episode on rental cars. "Oh, you're good at *taking* reservations... You're just not so good at *holding* reservations. And that's really the important part...the *holding*."

Yes, they're replicating continuously. That means if you fat-finger a document, your mistake could be immediately replicated onto the backup. The only backup that you will have is the last snapshot. That is not continuous protection; it's replication with snapshots—also referred to as near-CDP. A truly continuous product would restore your file right up to the point when you fat-fingered it.

They want to differentiate themselves from the "old" way of doing backups and draw attention to the fact that they're making these snapshots throughout the day — usually hourly. That's great—just don't call it continuous. It doesn't matter if the snapshots are being done once an hour, once a minute, or even once a second. Each of those would be called a time *period*, and *period* is an *antonym* of *continuous* in the thesaurus I use.

You either save every change, or you don't. If you're taking snapshots, you're not saving every change. It's as

simple as that.

This is not to say that I am not a fan of near-CDP products. Some of the coolest data protection things I've ever done have been with snapshots and replication. I also think that most people's requirements would easily be met by an hourly snapshot. I just don't want these products calling themselves CDP products.

It's like a Fibre Channel array calling itself NAS because it is storage attached to a network. Come on, people!

A CDP system has an unnoticeable backup window because it's copying only changed bytes as they change throughout the day. If they support the block-level recoveries discussed earlier, they also have incredibly fast RTOs. They likewise have infinitely granular RPOs because they can recover any file or filesystem to any point in time. This means that they can meet any type of synchronicity requirement, as they can recover 1, 10, or 100 systems to any synchronized point in time that you would like.

Different CDP products also back up different things. Some are filesystem-based, enabling you to back up and recover any files within that filesystem. Others are database-centric, providing CDP functionality only to a particular database, such as Exchange or SQL-server.

Unlike traditional backup products, file-based CDP products are not going to provide interfaces for your database applications, and believe they're unnecessary. Such vendors say that they copy blocks to the backup destination in the same order that they are changed on the client. They can therefore put the files back to literally any point in time that you want. Restarting your database after a CDP recovery causes it go into the same mode that it would go into if the server were to crash. It examines the datafiles, figures out what's inconsistent, rolls backward or

forward any necessary transactions or blocks, and your database is up. (By the way, if this crash recovery process didn't work, your database vendor would be out of business. Servers crash, and they have to prepare for that.) If the CDP product puts the blocks back in the same exact order they were changed, the database should be able to recover from any point in time. They also say that if for some extreme reason the database can't perform crash recovery from the 12:03:57:01 p.m. image, recover to 12:03:57:00 or 12:03:56:59. Some products can even present a logical unit number (LUN) or volume to your database that it can mount and test before you do the recovery.

Your database vendor may have a different opinion about CDP. They may feel that if you're not using their supported backup method, you shouldn't call them for support if something goes wrong. If you're considering a CDP product to back up your database, you should have that conversation with your database vendor and then make your own decision. In addition, your DBAs have to be sold on CDP as well. Some may think it's revolutionary; others will think it's scary. If you like the idea, keep pushing both your database vendor and your DBAs to consider it. Times change. It wasn't that long ago that Oracle didn't support NAS and snapshots; now it loves them.

Remote Office Backup

Figure 8-3 shows a remote office backing up to a central office using either de-duplication, near-CDP, or CDP software. If the clients on the left of the drawing are too large to meet their RTO if recovering from the central office, you can back up to a local recovery server that is used to facilitate nondisaster major recoveries. That server then replicates its backups to a centralized server.

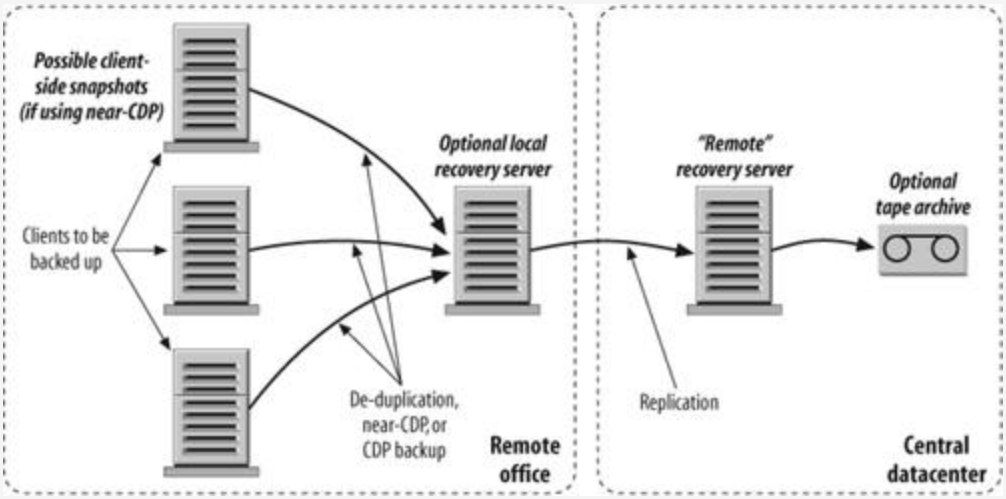


Figure 8-3. Backing up a remote office

---

[2] In another book of mine, *Using SANs and NAS*, I tried to coin the term “client-free backup” for a backup that used a different server than the client. It never caught on. I give up. The definition given here is the industry-standard one.

[3] Some vendors refer to split-mirrors as snapshots. I prefer to reserve the term snapshot for virtual copies. I would call the split mirror a business continuance volume, or BCV.

Simultaneous Backup of Many Clients to One Drive

There are now backup drives that can write much faster than many disk drives. Backup drives now use Fibre Channel and large buffers, and sometimes write multiple streams of data to the drive at once. The result is that many backup drives can write at 100 MB/s or faster. As mentioned in Chapter 9, the difficulty with most of these drives, though, is that they are *streaming* tape drives. This means that you must supply them with a steady stream of incoming data equivalent to their maximum throughput. If you don't, the drive does not stream, resulting in a significantly slower transfer rate than it is capable of. This problem, of course, affects only streaming tape drives. It does not affect backup drives that simulate a disk drive, such as magneto-optical, CD, DVD, or virtual tape drives.

All streaming tape drives are designed to write most effectively at their optimum speed. If they are supplied with data at a slower rate than that, the result may be surprising. What begins to happen is referred to as *repositioning*. See the section "Tape Drives" in Chapter 9 for more information on this concept. The drive spends most of its time repositioning and has less time to actually write data. The result is that it will write at a small fraction of its maximum data rate.

When would this happen? Consider a single-threaded backup process such as `dump` or `tar`. It encounters many potential bottlenecks before getting to the drive. The first obstacle is the disk itself, which may not be able to supply data at a sufficient rate. Another is the SCSI bus and system to which the drive is connected. It may be busy satisfying other I/O requests, hence impacting the data rate

that can be transferred across the bus. The next, of course, is the network. If it's a 100 MB network, the best possible data rate is less than 10 MB/s.[4] If it's a gigabit network, any individual server can typically handle only about 60–75 MB/s. (Hopefully this limit will go away soon.) The final possible bottleneck is the system to which the backup drive is connected. Any one of these bottlenecks could slow the data rate to less than the optimal speed for the backup drive. If that happens, a tape drive stops streaming and begins repositioning in order to keep pace with the incoming data. When this happens, it becomes the new bottleneck.

What can be done about this? Most commercial backup products solve this problem by implementing multiplexing. Multiplexed backups read data from several systems and several disks at one time and supply the data from all of these threads to the backup drive simultaneously. The data from these different sources is then interleaved onto the volume. The backup drive is therefore always being supplied with a sufficient stream of data so that it can write at its maximum speed. A cautious administrator learning of this feature for the first time might be worried. Some administrators don't like the fact that their data is spread out all over the volume. However, this feature is really the only way to stream a backup drive that can read or write at speeds faster than a disk. This is why most commercial backup products implement multiplexing.

Tip ✎

Different backup software vendors use different terms for this feature. It is sometimes referred to as *parallelism* or *interleaving*.

The downside to multiplexing is that it slows down the speed of restores. Therefore, multiplexing was considered by many to be a necessary evil. The feature that has replaced it is disk-to-disk-to-tape backup, where backups are first sent to disk, then sent to tape.

---

[4] Dividing megabits by 8 gives us megabytes, so 100 Mb/s translates into about 12 MB/s, but is usually closer to 10 MB/s in reality. (And the math is easier this way.)

Disk-to-Disk-to-Tape Backup

A very important methodology in today's backup systems is to first back up data to disk, and then copy it to tape. This is referred to as *disk to disk to tape*, or D2D2T. As long as the backup product can support automatically copying data from one media type to another, it can be used to support D2D2T backups. However, support for automating staging based on capacity can help a lot.

For example, suppose you back up to a large disk array or virtual tape library (discussed in detail in Chapter 9), and you would like backups to stay on that VTL as long as possible before being moved to tape. The ideal would be that the backup software would automatically copy the backups from disk to tape, and would automatically expire the disk-based backups to make room for more backups. This kind of functionality is often called *disk staging* and is usually only available if you are backing up to a filesystem (i.e., not using a VTL).

Different backup software companies support D2D2T in a number of different ways, and some of them are more fond of it than others. If you are a fan of D2D2T, you should definitely look into how well the backup software you are considering supports the features in this area.

Simultaneous Backup of One Client to Many Drives

How does a backup product get a very large database (VLDB) or very large system (VLS) onto backup media? This is a slightly different scenario from the one described previously. Multiplexing allows 5 or 10 systems to share the same device, so that backup devices are constantly streaming, and backups can be done in a timely manner. In contrast, the situation that calls for *multistreaming* is a single system that could not possibly back up its files to a single backup drive in one night. Even if one had a backup drive that is capable of 100 MB/s, that is only 720 GB per hour, or 5.76 TB in an eight-hour period—assuming the drive could be streamed for that long. What if the system contains 20 TB of data? What if the drive is capable only of 25 MB/s because of the network it's running on?

The only way to get that kind of speed is to use several drives simultaneously. (Other combinations would work, of course, but they all require multiple simultaneous backup drives.) However, in order to make that happen, the backup software needs to be able to take the one system and send it to many devices simultaneously. Many products are able to do this. What is important is how they do it.

The most common way that backup software companies tell you to back up a VLDB or VLS is to configure multiple backup definitions, each of which is a subset of the entire system, then run them simultaneously. For example, there could be one backup definition that backs up everything, excluding */home1* and */home2*. Then there could be a second backup definition that backs up just */home1* and */home2*. The software then would be told to back them up at the same time. That way, two different devices would be operating at

the same time—potentially cutting the backup time by as much as half.

This method is not desirable for a few reasons. The first is that it is difficult to figure out where to divide the partitions up, and you'll never get it exactly right. Even if you get it right one day, things will change. The second problem is that it requires you to maintain an include list that must be updated every time they add a new filesystem.

> **Warning ☠**
>
> Backups should never be defined this way. If backup definitions must be split up, always start with an "everything except" backup definition. That way, when a new filesystem is added, it will get backed up automatically.

Glad Nobody Asked for That One!

I was using a backup product that required me to create separate backup definitions if I wanted to do one-to-many. The host was so big and my tape drives were so slow that I had to define five separate backup definitions. First, I had a backup definition that included everything except */data1*-*/data8*. Then I had four more, each of which backed up two of the */data<x>* filesystems. When the machine's administrator added */data9*, it was backed up automatically by the "everything except */data1*-*8*" backup. However, when */data10* and */data11* were added, they didn't get backed up at all.

One day I noticed that I couldn't find any history for the */data10* filesystem. It was only after a frantic call to tech support that I found out the error was mine. When I excluded */data1*, what I actually said was, "exclude all

filesystems that match the regular expression */data1*." Unfortunately, */data1* also matches */data10*. For more than two months, there were two filesystems that weren't getting backed up. Luckily, nobody needed a restore. This is why I say to be very careful when excluding filesystems using regular expressions.

A better method is if the backup product automatically creates the multiple jobs for you. If they support this, they usually do it on the filesystem level, creating a job for each filesystem. One challenge with this is a single large filesystem. Due to modern advancements in filesystems, they can now be several terabytes. So now, not only are there several terabytes of data to back up from one system, but all the data resides in one filesystem.

There is no way to back up a multiterabyte system to a single backup drive in one night. The only way to get that kind of speed is to use several drives simultaneously, on four separate channels. However, in order to make that happen, the backup software must be able to take the one filesystem and send it to many devices simultaneously. As of this writing, this is an area that only two products have properly addressed.

Data Requiring Special Treatment

All commercial backup products can back up normal filesystem data. However, there is a lot of data that does not reside in a normal filesystem. Some data does reside in a filesystem but still requires special treatment before it can be backed up. Find out how the product that you are considering handles data such as this:

Network-mounted filesystems

: It is sometimes necessary to back up data that resides on a disk that is mounted from another machine.

Data that needs a custom script to back it up

: Some types of data fall in between “normal” filesystem data and database data. If this data is created by a special utility, that utility must be shut off prior to running the backup.

Relational Database (RDBMS) data

: Although some RDBMS data can be backed up using shutdown/startup scripts, there is now a much better way to back up most commercial database products.

The following sections discuss these three needs.

Network-Mounted Filesystems

Why would anyone want to back up via NFS? In almost every shop, there is a client that is not supported by commercial backup software. Perhaps it’s an older operating system that is no longer supported. Perhaps the software vendors aren’t convinced that the operating system has enough marketshare. (Remember that these

vendors aren't in this for free.) What good does it do to port your software to a platform that no one is using?

One solution to back up such a client would be to NFS-mount its filesystems to the backup server and back up the data via NFS. Yes, NFS can be a horrible way to back up. Yes, there are problems with restoring via NFS. But, as they say, something is better than nothing.

Along with NFS is Microsoft's CIFS filesystem, which uses the SMB protocol. This protocol works similarly to NFS, and a supported backup server could mount such drives over the network and back them up that way. (One of the problems with this method is that it must mount these drives to a Windows server, or it will not back up or restore the ACLs—but at least you'll have the data.)

The other issue with NFS- and CIFS-mounted filesystems is whether the backup software can exclude them. Imagine what would happen in most Windows environments if the backup product started to back up *all* CIFS partitions! Typically, this is avoided by excluding all NFS and CIFS mount points, but some products can selectively back up NFS and CIFS partitions. Some products allow you to configure a client in such a way that it backs up all CIFS/NFS-mounted partitions that it contains.

Custom User Scripts

At one point, custom user scripts were the only option available for backing up databases. The backup product would run a special program written by the administrator that shut down the database. The backup product then would back up the files or raw partitions on which the database resided. Finally, the program would restart the database. Depending on the size of the database and the

uptime requirements, this approach to database backups may still be a viable solution for some environments.

This method is now used for some types of data that do not fully qualify as “database” data but are dynamic enough that they require custom handling. Perhaps you use a network-monitoring tool that continually probes the network and stores the result in several interrelated files. Since those files must be backed up at a consistent point in time, the network-monitoring tool will need to be shut down while the backup is running. This can be accomplished by a custom script.

Databases

Several years ago, no commercial backup utilities backed up database data directly. The best you could do was to shut them down with a custom script as discussed in the previous section. However, databases are now so large that this option is no longer viable for many environments. Now all of the major database vendors have come out with some sort of Application Programming Interface (API) that commercial backup utilities can use to back up the vendor’s database. These utilities share four common traits:

- They can pass a data stream to a third-party utility, such as a commercial backup product. (Some also can perform standalone backups.)
- They can pass a data stream to a third-party utility while the database is online.
- They can provide multiple, simultaneous threads from the same database.
- Once set up, they can be called from either that commercial backup utility or the database interface. (For example, you can log in to the database server as

the DBA and issue a database command, and that will automatically start a session with the commercial backup utility. You can also issue a backup command from the backup product and have it call the database product.)

The question you must ask the backup vendor is, “Which of these databases have you ported to?” Even if you aren’t using any database products today, the number of databases that a particular product backs up to demonstrates its level of commitment to the enterprise market.

Also, it should be mentioned that some other interfaces do not use the vendor-supplied APIs. Additionally, some commercial backup utility vendors have independently written their own interfaces to these database products, and these interfaces do not use the database vendor’s API either. The preferred method should be to use the vendor-supplied API. Backups and restores done through some other method usually are not supported by the database software vendor.

Storage Management Features

Another important term to know is *storage management*. Just as some people think the word *Internet* was invented within the last few years, many think that storage management is a new concept. It actually goes way back to the mainframe days when 3480 tapes were much less expensive than disk. It became necessary to move important, yet unused, data off the expensive disks and onto less expensive 3480 media. (This is one way of *managing* the available *storage*, thus the term *storage management*.) SCSI disks were a lot cheaper, so Unix environments just bought more disks when they needed more storage space. Unfortunately, this “disk is cheap” mentality has led users to create far more and far bigger files than ever before. Within the last few years, though, IS managers have started to become very frustrated with the amount of money they are spending on disk. They believe that a number of files could be moved from disk to a slower storage medium without anyone noticing. This belief has given rise to a demand for storage management in the Unix arena. Just what is storage management, though?

This section examines the three principal storage management concepts that relate to backups:

- Archives
- Hierarchical Storage Management
- Information Lifecycle Management

Archives

Archives are designed for logical retrieval of information, that is, the retrieval of information grouped in a logical

way. For example, with archives you can store reference data such as:

- The CAD drawings, parts lists, and other manufacturing information for a widget your company used to make
- All of the information pertaining to a former customer
- All of the information related to a closed project, account, or legal case
- Tax returns, financial records, or other records for a particular year

In other words, information that can be grouped in a logical way can be archived and stored so that a company can retrieve it based on that logical grouping. Once a widget is no longer produced, a case is closed, or a tax year has passed, the information pertaining to that event or item just takes up space. We might need to reference it again for some reason, but we don't want it filling up our high-end storage, so we archive it and delete it from our tier-one storage.

The second way that archives manifest themselves is in the logical storage of active data. Suppose, for example, it was discovered that a critical safety part was removed from a particular widget's design. It would be important to see every version of the specification, along with information about who changed it. Also, consider the common practice of electronic discovery of email systems. Think about the discovery requests that can come from someone in management being accused of harassment or discrimination, a trader accused of promising financial returns, or a company charged with colluding with its competitors. Such accusations may result in e-discovery requests that look like the following:

- All emails from employee A to employees B, C, and D for the last year
- All emails and instant messages from all traders to all customers for the last three years that contain the words “promise,” “guarantee,” “vow,” “assure,” or “warranty”
- All emails that left a company going to domains X, Y, and Z or to certain specific email addresses

Using a backup program to create archive files isn’t a good idea: trying to find specific information in backups is costly and time-consuming.

> **Warning ☠**
>
> A bottle of grape juice left on the shelf long enough will ferment, but no one would call it wine. Similarly, it’s possible to retrieve data from old backups, but no one should call them archives. Simply put, backups make lousy archives.

Backups make lousy archives

The most common way data is archived is by keeping backups for a long time. Weekly or monthly full backups are performed, and then the backup is kept from 1 year to 50 years, depending on business requirements. There couldn’t be a worse way to archive.

Using backups as archives poses many difficulties. The most common use of backups as archives is for the retrieval of reference data. The assumption is that if someone asks for widget ABC’s parts (or some other piece of reference data), the appropriate files can just be restored from the

system where they used to reside. The first problem with that scenario is remembering where the files were several years ago. While backup products and even some backup devices are starting to offer full-text search against all your backups, the problems in the following paragraph still exist.

Even if you can remember where the files belong, the number of operating systems or application versions that have come and gone in the intervening time can stymie the effort. To restore files that were backed up from “Apollo” five years ago, the first requirement is a system named Apollo. Someone also has to handle any authentication issues between the backup server and the new Apollo because it isn’t the same Apollo it backed up from five years ago. Depending on the backup software and operating system in question, the new Apollo may also need to be running the same version of the OS and applications the old Apollo was running five years ago. Otherwise, there may be incompatibilities in the filesystem or database being restored.

Satisfy electronic discovery requests

Backups are also used to satisfy electronic discovery requests, which can be even more challenging. Let’s use the most common electronic discovery request as an example: a request for emails that match a particular pattern and were sent via an Exchange server. (The following also applies to other email systems, such as Lotus Notes or SMTP.) There are two big problems with using backups to satisfy such a request. The first is that it’s impossible to retrieve all emails sent or received by a particular person. It’s only possible to restore the emails that were in the Exchange server when backups were made. If the discovery request is looking for an email that

somebody sent, deleted, and then cleared from her *Deleted Items* folder, it wouldn't be on that night's backup, and thus would never show up when you attempted to retrieve it weeks, months, or years later. It would therefore be technically impossible to meet the discovery request using backups. This means that even after doing your best to successfully satisfy the discovery request, a plaintiff may claim that you haven't proven your case.

The second problem with using backups to satisfy an Exchange electronic discovery request is that it's very difficult to retrieve months or years of emails using backups. Suppose, for example, a company performs a full backup of its Exchange server once a week, and for compliance reasons, it stores these backups for seven years. If the company received an electronic discovery request for emails from the last seven years, it would need to perform many restores of its entire Exchange server to satisfy the request. The first step would be to restore the Exchange server to an alternate server using last week's backup. Next, you would have to run a query against Exchange to look for the emails in question, saving them to a *.PST* file. You would then have to restore the Exchange server using the backup from two weeks ago, rerun the query, and create another *.PST* file. It would be necessary to restore the entire Exchange server 364 times (7 years multiplied by 52 weeks) before you're done. And almost every step in this process would have to be done manually.

Accomplishing this isn't impossible, but the recovery effort will entail an incredible amount of time and money. A plaintiff in a civil suit or the government doesn't care how much it costs the defendant; your company has a court order to produce the emails—regardless of the cost. Yes, a good lawyer can argue against this, but it can be argued both ways. Do you really want to take that chance?

Other backup bugaboos

Backups are also an extremely inefficient way to store archives. While an archive system makes sure it has only one or two copies of a particular version of a file, a backup system usually has no such logic. If a company is using weekly full backups as archives (or creating “archives” with its backup product but not deleting the original files), and storing its archives for 7 years, it will have 364 copies of many of its files stored on tape—even if those files have never changed. This leads to an incredible amount of media waste.

Another strike against using backups as archives is the number of times a company changes backup formats and tape formats over the years. Almost every company using backups as its archives has a number of older tape and backup formats it must continue to support for archive purposes. While older tape formats can be converted with a lot of copying, converting older backup formats is another story. Most people choose to hold onto both old tape formats and old backup formats, and hope they never actually have to read them.

True archiving

The most important feature of an archiving system is that the archive should contain enough metadata to allow information to be retrieved in logical ways. For example, metadata can include the author or the business unit that created an item. (An item can be any piece of archived information, such as a file, a record from a database, or an email.) Metadata might also contain the project the item is attached to or some other logical grouping. An email archive system would include who sent and received an email, the subject of the email, and all other appropriate

metadata. Finally, an archive system may import the full text of the item into its database, allowing for full-text searches against the archive. This can be useful, especially if multiple formats can be supported. It's particularly expedient to be able to do a full-text search against all emails, Word documents, PDF files, etc.

Another important feature of archive systems is their ability to store a predetermined number of copies of an archived item. A company can then decide how many copies it wishes to keep. For example, if a firm is storing its archives on a RAID-protected system, it may choose to have one copy on disk and another on a removable medium such as optical or tape.

Two types of archivers

Archive systems can be roughly divided into two categories depending on the way they store data. The first is the traditional, low-retrieval archive system that's attached to your backup software package. Such an archive system allows you to make an archive of a selected group of files; attach limited metadata to it, such as "widget XYZ"; and then have the archive system delete the backup files in question. The good thing is that it allows the attachment of metadata and can reduce multiple copies in the archive by deleting the duplicate backup files as they're archived. The bad news is that if you want to search archives using different types of metadata—such as owner, time frame, etc.—you may need to create multiple archives. The main use for this type of archive is to save space by deleting files attached to projects or entities that are no longer active.

The second—and newer—category of archive systems realizes that any archived item might need to be retrieved for different reasons and would thus require different metadata. To support multiple types of retrievals, it's

important to store the actual archived item only once but to store all of its metadata in a searchable database. Such a system realizes that a given archived item might be put into the archive not to save space, but to allow it to be searched for logically. Therefore, unlike its predecessors that stored the only copies of reference data, newer archive programs store an extra copy of the data, leaving the original in place.

As discussed previously, one of the problems with using backups as archives is that they won't have all occurrences of a file or message; they'll have only those items that were available when the backup was made. Some of the newer archive systems solve this problem by archiving data automatically. For example, every email that comes in or is sent out is captured by the archiving system. Every time a file is saved, a version of the file is sent to the archive system.

Another advantage of newer archive systems is their use of single-instance store concepts. They store only one copy of a file or email, no matter where it came from or who it went to. (Of course, the archiving system records who it came from or who it was sent to.) If that file or email is then changed and sent/stored again, the archiving application stores only the changed bytes in the new version. Single-instance store saves a lot of disk space.

Regarding the format issues of backups as archives, many archive systems still grapple with those issues. Many people still store their archives on tape, and as time passes, people may change their archive software. Therefore, this problem could persist even in archives.

Newer archiving systems also serve as a hierarchical storage management-like system, automatically deleting large, older files and emails, and invisibly replacing them with stubs that automatically retrieve the appropriate

content when accessed. This is one of the main business justifications used to sell email archive software. In addition to satisfying e-discovery requests, you can save a lot of space by archiving redundant and unneeded emails and attachments.

Surveys show that more than 90 percent of typical email storage is consumed by attachments. If you can store only one copy of an attachment across multiple email servers (and Exchange Storage Groups), and replace it with a stub, you can save a lot of storage. If you add delta-block incrementals to that, you can save even more storage.

Tip ✎

While Exchange does single instance storage within a storage group, it does not do so across multiple storage groups or multiple servers. Suppose you send an email to four people, two of whom are in the same storage group on the same server, one of whom is in a different storage group on the same server, and one of whom is on a completely different Exchange Server. Exchange would store one copy of the email in the storage group with the first two users, another copy in the storage group with the third user, and another copy in the second Exchange Server with the fourth user. A product like the one described here would take all three copies out, store one copy on the archive server, and put a stub in place so that any user accessing the email would retrieve it automatically from the archive.

If your company has more than one employee, it wouldn’t be hard to build a business case for archiving. And if you’re using backups as archives, you could be in for a pretty rough time when you get an electronic discovery request.

Perhaps you should look at an email archiving product or an enterprise content management product today.

Hierarchical Storage Management

The second principal storage management concept is Hierarchical Storage Management, or HSM. HSM is a horse of a different color. While archiving allows someone to delete files from disk once they have been archived, that is not its primary purpose. Its primary purpose is to make those files easy to find years from now when the user doesn't remember what system they were on. HSM's primary purpose is to automatically monitor filesystems, looking for files that meet certain conditions, such as a file that has not been looked at for a long time.

In a truly hierarchical system, this would involve successively less expensive media types. For example, the file might be moved from a high-speed, high-availability system to older, nonmirrored disks. It then might be moved to optical disks and eventually to tape. Each of these levels is less expensive than the previous level but also has a longer access time. When a file is moved, or "migrated," to a less expensive level, the HSM system leaves a *stub* file behind that has the same name as the original file. If anyone attempts to access the stub file, the original file is retrieved automatically by the HSM system. This is invisible to the end user, other than the extra time taken to access the file. (Some high-speed systems can reduce this time to something that a typical user would never notice.)

Consider a heavy CAD environment. Engineers may make new drawings every day but want to keep the old ones around for reference. Suppose somebody asks five years from now, "What parts went into XYZ?" Without an HSM system, the drawing for XYZ would have to remain on disk

for years, simply taking up space. Now, if the engineer making the drawing was conscientious about the disk space she was using, she might contact the administrator and ask for her files to be archived. (However, most end users are not concerned about the administrator's disk space problems.) An HSM system would proactively monitor this CAD directory and "notice" that this particular group of CAD drawings hasn't been examined for more than a year. It then would migrate them to a less expensive storage medium without any action from the engineer. When the engineer did eventually need this file, all she would have to do is open it up in her CAD application. It would be retrieved automatically. To her, it would appear as if the system was really slow that day. If she had been educated about HSM, she might think, "Oh, the file must have been migrated." As long as the file comes back, she probably won't mind the HSM system at all. In an HSM system, there does need to be some education of the user community; they need to know that their files are being migrated. If they do not know, they will call the help desk whenever they experience a delay when retrieving a file.

Warning ☠

Implementing an HSM system should be done slowly and methodically with significant help from someone who has set up such a system somewhere else. Unrealistic migration policies and improper end-user education can spell disaster for an HSM system. Many system administrators have been required by their end users to remove or deactivate their HSM systems. HSM *can* work, but so many people have seen poorly configured HSM systems that they have earned a really bad reputation. Step very carefully when moving toward an HSM solution.

Information Lifecycle Management

Information Lifecycle Management (ILM) is more of a concept than a technology. Where HSM and archiving systems typically assume that a file gets less valuable as it gets older, an ILM system recognizes that different data have different values over time, and the value of a piece of data may go up and down several times.

The best example of a pattern that an ILM system would recognize is the exploration data for a oil-and-gas company. They spend years searching and searching for oil. Then they've got some information that some oil is in a particular place. As they prepare for permits to drill in that area, the data is highly valuable. While they wait for approval, the value of the data goes down. Once they get approval, the value of the data goes right back up. Once they start drilling, they'd better not lose that data, or they can cost themselves millions of dollars in lost drilling time. Once the oil has been extracted and there's no more in that

particular place, the data is all of a sudden no longer important again. It sits there for some indeterminate amount of time until someone decides to file a lawsuit claiming they drilled in the wrong place. Voila! The data is important again, as is the data surrounding it, such as permits, drilling records, etc.

This is an extreme example, but it illustrates the important concept behind ILM. What matters is what the lifecycle is of *your* data. Ask your backup vendor how they will help you manage that lifecycle.

Reduction in Network Traffic

One of the best ways to perform backups is to install a completely separate network to carry all the backup traffic. However, many people do not have the funding necessary to set up such a network, so they need to be careful about how their backup system influences the network that it shares. There are a few different things that backup products can do to minimize their impact on the network, whether it is a private one for backups or the corporate backbone.

Keep Backup Traffic at the Subnet Level

One of the best ways for a backup product to reduce the amount of traffic that goes between networks is to distribute the backup devices to the subnet level. Keeping backup traffic on its local subnet results in two benefits. First, backup traffic is distributed, so that no single subnet sees all the traffic. The overall impact to the internal customer (who also is trying to use the corporate intranet) is therefore reduced. The second benefit is that the overall throughput of the backup system will be able to scale with the network as the network grows by the addition of subnets. Consider the drawing in Figure 8-4. There are six different subnets (100.10–100.15), each of which is switched 10 MB (switches not shown). Each of the switches is plugged into a router, which also is plugged into the 100.1 subnet, where the backup server resides.

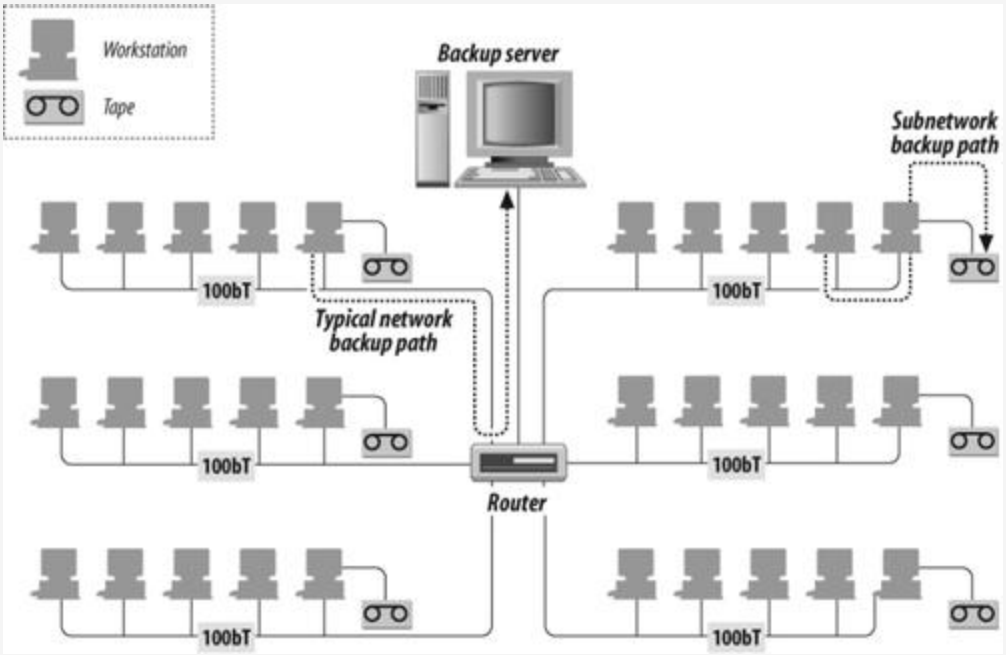


Figure 8-4. Network backup data paths

A typical, network-based backup system causes data to flow in the manner illustrated by the data path on the left. The data path represented by the left-side dotted line shows that the data goes over the 100.10 subnet, the router, and the 100.1 subnet to the backup server. If all six subnets are backed up that way, the total backup throughput could be no higher than the speed of the 100.1 subnet or the router because all traffic must go through them. The result is that the backup system never streams the backup drive.

However, if the backup software supports remote devices, the backup traffic can be localized to the subnets, as illustrated on the right. Even without taking advantage of switching technologies, the backup system now has an overall throughput of 6 MB/s instead of 1 MB/s.

Use Client-Side Compression

Backup software products also can use client-side compression to reduce network traffic. This means that

files are compressed on the client before they are sent across the network. This is very CPU-intensive on the client end, but the amount of data that actually goes out on the network can be greatly reduced. If the system CPUs are fast and the network is slow, this can improve overall backup throughput and reduce total backup time.

> **Tip ✎**
>
> Using client-side compression can slow down restores as well. How much restores are affected varies with the product and the nature of the systems being backed up. Before using compression, test to see how much it affects restore performance. It may be too great a sacrifice.

Incorporate Throttling

Another feature that is difficult to find in backup software products is called *throt tling* and can help reduce the effect of the backup traffic on the overall network. Some products allow the administrator to throttle a backup client, that is, to specify how many megabits per second a backup client can send out on the network. That way, even on a 100 Mb network, the backup system would use only a prescribed percentage of the total bandwidth, since that's what it was told to do. This slows down the backups, of course, but the people trying to use the network will surely appreciate it.

Storage Area Networks

Storage area networks, as shown in Figure 8-5, present several advantages when used with a backup system. A SAN:

- Provides scalable storage devices, which can be shared regardless of platform
- Grants faster data migration, allowing storage redeployment in a highly available, distributed, or centralized environment
- Supports higher levels of I/O throughput
- Allows increased performance on backup and restore tasks
- Contributes to better host connectivity, due to offloading the LAN from storage access-related traffic
- Promotes sharing of both storage and backup devices across the enterprise

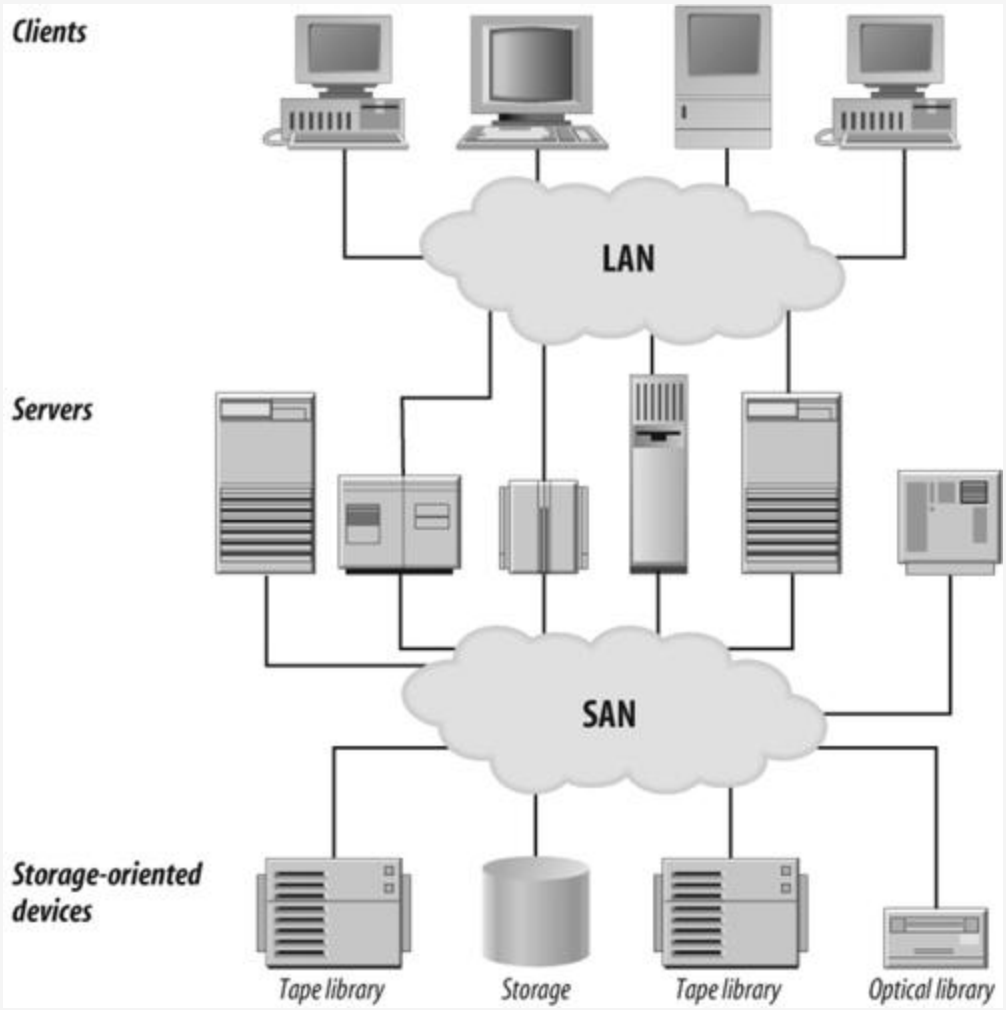


Figure 8-5. Connecting any type of peripheral via a SAN

When considering SAN technology in light of the backup and recovery industry, it is the ultimate network traffic-reduction method. It allows SAN-connected hosts to perform LAN-free backups (see the section "LAN-Free Backup" earlier in this chapter), taking advantage of the recent advancements in CPU and memory usage reduction. It does all this while allowing maximum utilization of large libraries. If you have a large environment, and especially if you have very large servers, you definitely should find out which backup applications support LAN-free backups.

Support of a Standard or Custom Backup Format

A *custom* backup format is one that is readable only by a particular commercial backup utility.[5] Backups made in a custom backup format cannot be read by native backup utilities such as `tar`, `cpio`, or `dump`. Some backup products use a custom format that is published or freely available. Although their backups can't be read with native utilities, a programmer theoretically could write a program that would read their backups. Some backup products are completely proprietary. In fact, some products are so proprietary that even their own product can't read a volume if the indexes for that volume have expired or been deleted. A standard backup format would be a format that is readable by a standard utility. There are two schools of thought on this subject.

There are those who feel that using proprietary or custom backup formats is dangerous. If the backup volumes can't be read by native utilities, what do you do when the commercial backup product is broken? They prefer to use utilities that back up using industry-standard backup formats such as `cpio` or `tar`. They provide a sense of security that is just not possible when using a custom backup format. Companies often switch backup products, and when that happens, their old volumes are not readable by the new product. If the volumes were readable by standard utilities, however, they still could be used for restores. What is commonly done in this scenario is to keep the old backup system running for restores only.

> **Tip ✎**
>
> Vendors are on both sides of this debate. What follows is my best effort to explain the pros and cons of each side. You have to choose which pros and cons are most important to you.

Some say that old backup formats are just that—old. They served their purpose, but it is time to move on to more sophisticated utilities. There are two problems with native utilities. The first problem is that they are always changing. The `dump` command is filesystem-specific, and different versions of `dump` are incompatible. The `tar` and `cpio` commands also have changed their formats over time and are not always compatible between different operating systems. (`ntbackup` has remained the same over the years.) The second problem is that native utilities have significant limitations, such as pathname lengths and the inability to handle open files. The most significant limitation, though, is their inability to generate or receive multiple backup streams.

> **Tip ✎**
>
> One of the open-source projects covered in this book—Bacula—chose to create its own custom format using this same logic.

Standard Backup Formats

On the other hand, longtime system administrators have learned how to use `ditto`, `dump`, `ntbackup`, `tar`, and `cpio`. There is a familiarity there that just isn't going to be

possible with a unique format. Also, some people have been burned by commercial utilities that have come and gone. There have even been a few that have changed their own formats, making their old volumes unreadable by a new version of their own software! This means that concerns about custom and proprietary formats are valid. Since restricting yourself to native utilities significantly reduces the number of available products, make sure that you properly examine the limitations of these native utilities before doing so. The limitations of each of these utilities are discussed in the following sections.

The dump utility

The `dump` command is a Berkeley contribution and as such is not always included on some pure System V Release 4 systems. (It sure surprised me the first time that happened!) `dump` backs up a filesystem via the raw device, not through the filesystem. It therefore must know the structure of the filesystem that it is backing up, so each new type of filesystem requires a new version of `dump`. Also, a `dump` backup of one filesystem type will not necessarily be readable by the `restore` utility of another filesystem type. There have been a number of new filesystem types over the years. Each new filesystem usually comes with its own version of `dump`, and many of the newer versions are not reverse compatible with the older versions. (See the section “Different Versions of dump” in Chapter 3.) Some new filesystem types don’t come with a traditional `dump` command at all.

A backup tool should not rely on a native utility that changes from filesystem to filesystem. The backup volumes are not compatible between platforms, and even within the same platform, such as `(efs)dump` and `xfsdump` on SGI. Also, `dump` is not always available.

The tar, ditto, and cpio utilities

Unlike `dump`, `ditto`, `tar` and `cpio` access files through the filesystem just as a user does. Since they are not filesystem-dependent, they change much less over time than `dump` does. This may be `ditto`'s, `tar`'s, and `cpio`'s greatest advantage. However, there are different versions of `ditto`, `tar`, and `cpio` for each platform, and not all of them are compatible. It also should be noted that most of the commercial backup products that write in a `tar`- or `cpio`-compatible format do not use the actual `tar` or `cpio` command; they have their own command that writes in a format that is readable by `tar` or `cpio`. (This is the way `ditto` works.) That way, the commercial product can overcome some of `tar`'s and `cpio`'s limitations, such as `cpio`'s 255-character limitation and `tar`'s 100-character limit on pathnames. (GNU `tar` also has overcome some of these limitations; it is covered in Chapter 3.)

Custom Backup Formats

As stated earlier, the camps are divided into those backup products that use a standard format and those products that do not. Further, products that do not use a standard format should themselves be divided into two groups—those that publish their format and those that do not. Theoretically, a programmer who knows the format of the volume could write a program to read it. Most products depend quite heavily on a database that tracks the location of each file or piece of file on the volume. If the database is corrupted or lost, they may not be able to read the volume at all.

> **Tip ✎**
>
> Be sure to read the section “Content-awareness” in Chapter 9.

Should someone use a product that has a custom backup format? Before purchasing such a product, be sure to ask a few questions. Is the format of this volume completely proprietary, or is there a document explaining how it was written? Is there a standalone utility that allows me to read these volumes even if the catalog is down? If this product made a volume but then later did not know what was on it, could it reread the volume and determine the file sets that went to that volume?

Some backup programs that use custom formats come with a standalone utility that can read the volume without the use of the backup database, providing essentially the same functionality as a native command. This is a beautiful thing, but it’s harder to come by than you might imagine.

What happened to SIDF?

Some readers may remember the System Independent Data Format (SIDF) that was first proposed back in 1993 as an international volume-interchange format. It was used on a limited basis by a small number of backup products. If a product followed this format completely, not only would it have completely platform-independent volumes, but its volumes would be readable by other backup software products. The format barely gained acceptance. Any questions on the status of SIDF are answered by going to http://www.sidf.org: “*www.sidf.org* not found. Please check the name and try again.”

A Reality Check

Suppose you had a bunch of volumes that were written in `tar` format, and your backup software has been keeping track of them all. If that software is not functioning properly, how will you know what is on the hundreds, or even thousands, of backup volumes that you have? I suppose you could do a `tar tvf` on all of them and create your own "minicatalog." That's not an easy task. Suppose you had 500 or so tapes. It would take you more than a month to read them all. This is just to get a table of contents of these volumes.

A much better solution would be to get a backup system that you trust. Learn how to check the database for inconsistencies. Run those checks every day, and if any inconsistencies are found that can't be fixed, recover the database back to the point in time before it became corrupted. If the backup software allows it, you then have it reread any volumes that have been written to since then.

---

[5] The word *custom* is more appropriate than *proprietary* because *proprietary* implies that you are not allowed to know the format.

Ease of Administration

If a person is administering a relatively large backup system, the activities in the following list are performed all the time. How easy is it to do these things with the product you are considering?

Making duplicate volumes for off-site storage

> This feature allows an administrator to make copies of volumes and send the copies off-site. (Actually, a better method is to send the *originals* off-site. Doing the day-to-day restores from the copies is a constant verification of the copy process. Unfortunately, many products won't allow copies to be made this way.) The right way to do this is to copy from volume to volume, instead of running another backup. The copied volume should have its own identity. That way, the copied volume can be tracked in the catalog. (Older methods of copying volumes resulted in two copies of the same exact volume. The second volume never actually existed in the product's database; it was just another copy of the first volume. The same is true of some VTLs that replicate their data.) Some products also allow an administrator to specify the location of the copy volumes so that he knows which volumes are on-site and which volumes are off-site. That way, the software won't ask for the volume that it knows is off-site, if it knows it has another volume on-site that has the same data on it.

Copying an individual backup

> There is a downside to copying volumes. Depending on the level of volatility of the data and the luck of the draw, the system may or may not fill up a complete volume each night. If copies are being made and sent

off-site every day, what happens to the volume that is half full? If it is copied and sent off-site, the software cannot continue to write to the original volume, even though it has more room available. If it did, it would have to copy the entire volume again. Wouldn't it be better if the software knew how to copy last night's backups, regardless of which volumes happened to receive them?

Individual backup copying does just that. It is the ability to say, "take all of the files that got backed up last night (the backups) and put them on these volumes over here, which I will send off-site." Depending on the complexity of the software, last night's backups may have ended up as 100 MB pieces on 20 volumes. Backup cloning would take those 20 separate 100 MB backups and put them all on one 2,000 MB (2 GB) volume. This would be done after the backup, and the data traffic would stay local to the backup system—so it wouldn't affect the other CPUs, the network, or the users.

Simultaneous copying of backups

A very new feature that has begun to appear is the ability to copy the backup as it is being made. If things are set up right, and the software does what it is supposed to do, you actually can make copies in no more time than it takes to make the first backup. The better products would allow the two backup volumes to be different sizes, allowing for differences in compression and different types of media. (Perhaps you may want to put your on-site backups on one type of media and your off-site backups on another type of media.)

Consolidating a host's backup

This nifty feature of some backup programs allows for the consolidation of all of the backups for a given host

onto a volume or small set of volumes. This makes preparing for a disaster much easier.

Consolidating volumes that have very little data on them

This is a volume utilization feature. Due to different expiration times of different backups, there may be a large set of volumes, each of which has only a few hundred megabytes of data that are still useful. Volume consolidation consolidates these backups from all of the partially used volumes onto one volume, allowing the source volumes to be reused.

Filesystem/database discovery

Will this product automatically discover all filesystems/drives, or do you have to specify them manually? The latter requires you to constantly update the list of filesystems to be backed up—not good. Also see if it supports this feature on any databases. It sure is nice not to have to maintain the list of SQL Server databases to be backed up.

Excluding files

There are always files that should be excluded from backups: temporary Internet files, files in */tmp*, spool files, core files, etc. The types of files that need to be excluded vary from site to site. Be aware that different products have different methods for excluding files.

Interface types

Most products have Windows, Java, HTML, Motif, Curses, and/or command-line interfaces. Only a few have native Mac interfaces. Make sure the product has an interface that is appropriate for your environment. (I also should state the importance of an optional command-line interface in almost any environment; you can’t always count on a graphical display during a

disaster recovery, and you also may wish to do your work over a remote connection.)

Notification

How should the administration team be notified when a backup fails or needs attention? There are a number of different methods. The most common method is email. Some vendors even allow different email recipients for different types of messages. Other vendors allow you to write custom interface programs that send reports wherever you would like. Some backup products can even send alpha pages if there is a modem available.

Perhaps the most sophisticated notification method is the use of an SNMP trap. SNMP stands for Simple Network Management Protocol and was originally developed for network hardware so that routers from different vendors could understand one another. SNMP has been expanded to include all types of network monitoring, and it is what most of the commercial network monitoring tools are based on.

Installing

Most server installations are fairly automated. The real bear is the installation of client-side software. There are usually two ways that client-side software can be installed: manually or automatically. When using the manual method, you first must get the client software to the machine. This is done either by `ftp`, `nfs`, or `CIFS`. Then, run the installation program. Some products can completely automate client installation under certain circumstances.

Upgrading

The product is installed once, but it will be upgraded once or twice a year. Therefore, ease of upgrading is another very important feature to consider. Upgrading

also may be performed manually or automatically. However, some products (once installed) are able to update the software automatically. Since updating the software on hundreds of clients is the most difficult repetitive task you will do, this feature is quite handy.

Security

Historically, backups and security have had almost completely opposite goals. Things such as *.rhosts* files in Unix systems were absolutely necessary to gain any type of backup automation, and yet they are a well-known security problem. Fortunately, most modern backup products have worked around these problems. Here are some security issues to consider:

Daemon/service communication

: Any decent product is going to have a secure method of communicating with the client. They will not require insecure communication methods such as `rsh`.

System authentication

: How will the server and client verify each other's identity? Will they simply use hostnames and IP addresses? That's easily faked and should be avoided. Another method is to use the root password of the client. This requires the backup administrator to know the root passwords of every client—also not a good idea. Other systems use sophisticated one-time passwords that are very strong. Investigate how your backup software authenticates its systems.

User authentication

: How does the backup software authenticate administrators of the system, or people who want to perform user-level recoveries? Do they have their own database? Do they integrate with Active Directory or NIS?

Role-based authorization

Once a user is authenticated, are they all-powerful, or can you assign certain tasks to some people and not others? It would be nice if the person responsible for monitoring backups is not the same person setting them up. It would be nice to limit the number of people who can delete or overwrite backups.

Encryption

http://www.privacyrights.org maintains a list of privacy breaches, and as of this writing, it lists over a dozen tape-related privacy breaches. With all the attention these incidents are getting, more and more people are asking for encryption. Encryption can be accomplished in one of three ways. The data can be encrypted in the original filesystem. It can also be encrypted by the backup software as it's being transmitted to the server. Finally, it can be encrypted by a hardware appliance. If you're considering backup software encryption, make sure to talk to your backup vendor about it.

Ease of Recovery

I often quote an old coworker of mine, “No one cares if you can back up—only if you can restore.”[6] Ease of recovery and speed of recovery often are overlooked when evaluating backup products. Many small factors can make doing restores either very easy or impossible:

Platform independence

> This is a very important factor. Some products have gone to the trouble to ensure that every volume made by every version of their software can be read by every other version, no matter what platform it was made on. If this has been done, and a client is destroyed, its data still can be recovered, even if the replacement version is not of the same type. If volumes are not platform-independent, an administrator might need to keep a functional machine of each operating system version just for restores.[7] Having true platform independence also makes doing regular restores much easier.

Parallel restore

> This can be a very nice feature. When restoring a large directory or filesystem, the backups for that filesystem may be spread out over several volumes. Some products are able to read all these volumes at once, actually making the restore faster than the backup. When investigating this possibility, you also should find out if these volumes can be loaded in any order.

User restores

> Some environments have sophisticated users who like to be able to do their own restores. If this is true in your environment, this feature will come in handy. Those who

do not want users doing their own restores will want to know whether this feature can be disabled.

Relocated restores

This is *very* important. You sometimes need to restore a file that was originally located on another system. This different location may be a different host or a different directory. Some products do not allow this.

Bare-metal restores

A bare-metal restore is restoring a system from scratch, without even having a functioning operating system. (See Part V of this book for more information about bare-metal recoveries.) This is sort of the Holy Grail of backups—the ability to restore an entire system from nothing. Several products now offer bare-metal recovery for one or more platforms.

Multiple versions

This is also very important. A lot of backup products not only track the most recent version of a file that was backed up but also track all versions of the file that are on backup volumes. Sometimes it's necessary to restore a file to the way it looked four days ago.

Tracking deleted files

This one surprises many people. Suppose there is a filesystem that changes quite a bit. New files are added and deleted every day. (A good example of this would be where Oracle's archived redo logs go. Hundreds of files may be added and deleted every day.) When asking the backup software to restore this filesystem, you might expect it to restore the filesystem the way it looked yesterday. Unfortunately, many products will instead restore all files that were ever located in that directory! It takes extra effort on the part of the software to

“notice” that a file has been deleted and to *not* restore it unless told to do so. Failure to track deleted files can make restoring some filesystems very difficult.

Overwriting options

Has a user ever called and said that he blew away half his home directory because he typed `rm -r *` by accident? This user doesn’t want the program to blow away everything in his directory by restoring on top of it. A good way to protect against that is to tell the backup software to “restore everything in here, except those files that are newer than what we have on backup.” There are a number of other overwriting options, such as unconditional overwrite, prompt before overwrite, and don’t overwrite the same exact file.

---

[6] Ron Rodriguez gets credit for this one. Replacing Ron as “backup boy” was my first job in this field. I have no idea how many times Ron said this, but I guess it sunk in.

[7] Believe me, I know! I know of one place that still keeps around one or two AT&T 3b2s running SVr3 because of all the old `cpio` backups made on QIC drives that are unreadable on other platforms.

Protection of the Backup Index

The backup database, catalog, or index keeps track of which files were backed up to which volume. Since the backup system can't restore anything without this index, it becomes the single most important database in your environment. It is also the single point of failure in any backup system. As mentioned earlier, even if a volume is made with a format that is readable by a native utility, you still need the index to know what's on it. The backup index is the greatest invention since someone created volume labels, but if it goes bad, you are out of luck.

I Followed the Procedure

I had an experience at a gaming company that ran about 10,000 slot machines on a complete Windows shop and backed it up with a commercial backup software product. Because of a known, but unfixable, firmware issue with the library Fibre Channel cards, the library would lose a drive every now and then and corrupt the robot database. Somewhere early in the project, someone had written a document that detailed a fix for this error that had to do with uninstalling and reinstalling the backup software. Unfortunately, they left out a step: saving a copy of the catalog. This procedure was followed twice before it was caught within the first year of this datacenter's opening. Luckily, needed restores were run from a disk-based backup taken outside of our commercial backup software.

Brian Sakovitch

Backup indexes usually are located on the central backup server, but they can be spread out to what is sometimes called media or device servers. (A *media server* or *device server* is a server that is allowed to have backup devices.) One of

the first questions you might ask is, “How big will this thing get?” The typical answer is .5 percent to 1 percent of the amount of data that is being backed up. That answer is very misleading, completely wrong, and totally irrelevant. The total size of the data that is being backed up has absolutely nothing to do with the database size. Let me state that again.

> **Tip ✎**
>
> The total size of the data that is being backed up has absolutely nothing to do with the size of the backup database. It is the number of files being backed up, not their size, that determines the size of the database.

Each file that is backed up becomes a record in the index. That record will be the same size, regardless of how big the file is that was backed up.[8] The appropriate question, therefore, is, “How many bytes does each new file add to the index (a) the first time it is backed up and (b) any additional times it is backed up during incremental backups?” This number can then be multiplied by the number of files that are being backed up. This will show how big the index can get from one full backup. Multiply that number by the number of full backups the system is required to keep online. Using an estimate of a 2 to 5 percent daily volatility rate,[9] estimate how big the index will grow from each incremental backup. Multiply that number by the number of incremental backups that the system is required to keep online. Add that to the first number, then multiply by 2. The result will be a pretty realistic, albeit slightly exaggerated, estimate of how big the backup index will get.

Managing the growth of the index is also a big issue. Whatever database format they use, one of the index files

may grow larger than what the filesystem permits. If that happens, the index may get immediately corrupted. The backup product should have some method of dealing with this problem. Also, the entire index may get larger than the largest filesystem allowed, so it should be able to spread that data out across multiple filesystems.

Just as the volumes should be platform-independent, so should the backup index. You should be able to restore it to any system in which the server software runs and continue working. In order for this to work, the index needs to be completely platform-independent. Some products are, and some aren't. Some of them are not platform-independent, but they do provide a utility to move the index to other platforms. One of the best tests of this is to attempt to recover a Unix server's backup index to a Windows server.

Before committing to buying a commercial backup product, test its index restore procedure. Some products can restore the index in a single step, while others require 20 pages of steps. Once you actually purchase a product, test that procedure again. Then test it on a regular basis so that you never hear yourself saying, "My whole world just crashed, including my backup server. Now what am I supposed to do again?"

A couple of minor (but nice) features also are helpful. The first is the ability to change a client's name within the index. If a backup client's hostname changes, and the backup product does not support this feature, there are only two choices. The first choice is to give up all backup history for that host. The second is to pay for another license, because the software will recognize the new hostname as a new client.

Another very nice feature, seen as essential by some, is the ability to reread a volume back into the index. Suppose there is a volume that has been set aside and is now

expired out of the index completely. What if the only backup of a file that you need is on that volume? What if you don't know whether it's on that volume? Some products can perform the restore without having to reread the entire volume, while others can read the volume right back into the index, making it appear as if it were just backed up. Some products are not able to reread that volume at all! One factor that goes into the product's ability to read a volume like that is whether the vendor puts a copy of the index information onto the volume. It's an extra step, but I think it's well worth it. Basically, after every backup, the new portion of the backup index that was created from that backup is placed on the volume. That makes rereading the volume from scratch much easier. It is possible to reread a backup volume without placing the index on the volume, but having the index there makes it easier.

The importance of the backup index, and your familiarity with how it works, cannot be overemphasized. It is the lifeblood of any backup system and should be treated like gold.

Index Horror Stories

Most of my backup horror stories stem from problems with the backup index. My first bad experience was with backing up a large NFS server that was used to store home pages for a large online service's web servers. There were more than three million small files, which made the index so large that it would often become corrupted. Even after distributing the index over multiple slave servers, the size would still cause index corruption. As if regular index corruption weren't enough, we often would not catch the corruption until several days later when the backup product would act stranger than usual. Since we were foolish enough to keep only two days'

worth of index backups, we could not recover a reliable index. Eventually, we ended up dumping the index into ASCII files daily and then backing up those files from a different server with the regular retention schedules.

My other index horror story comes from the same site. In another effort to keep our index small, we stored the index of a backup for only two weeks (even though the data was kept on a backup volume for two months). I had one user who on multiple occasions deleted data after he was done with it, only to determine two and a half weeks later that he really wasn't done with it. Since the records containing those files had expired out of the index, every volume that might have the data had to be reread. One of those restores had me rereading more than 40 DLT 4000 tapes (in a jukebox that held only 28 tapes) while still trying to do regular backups. It took me more than three long days to read the tapes; even then I was not able to retrieve all of the data. Fortunately for my job, it was not mission-critical data.

Bryce Wade

---

[8] Some products do use a variable-length record so that things like the length of the pathname can slightly affect the size of the record, but the size of the file still has no bearing.

[9] This is actually a huge volatility rate, but most environments don't have any data on the *number* of files that change each day. Even if they've been monitoring their backup software, most reports talk only about how much data was backed up, not how many files were backed up.

Robustness

Horrible things happen to backup systems. Systems reboot and get powered off. Backup drives hang, libraries jam, and networks die. The “robustness” of a product can be measured by how well it deals with these sorts of problems. Can the product reroute backups from a failed backup drive to a good one? Will it even notice that a backup drive or process is hung? Is it able to recover from a client rebooting while the client is being backed up? Will this client rebooting in the middle of its backup corrupt the index?

These are very important considerations. Things will go wrong, and when they do, you want to know the backups are still OK. If the backup product is able to reroute around failures, retry open files, and restart failed backups, the worst that should happen to you is a really long report when you come in the next morning.

Automation

There are some very nice tape and optical libraries out there now. They have bar codes, automatic cleaning, hot-swappable power supplies, field-replaceable drives, and more. How well does this product take advantage of these things? One of the greatest features of using a modern library using bar-coded volumes. Put 20 volumes in the library, and then tell the backup software to read the bar code and electronically label the volumes according to what the bar code says. That sure beats swapping 20 volumes in and out of backup drives for a half-hour while they get labeled!

Volume Verification

Another often-ignored area of data protection software is its ability to verify its own backups. There are plenty of horror stories out there about people who did backups for years or months assuming that they were working just fine. However, when they went to read the backup volumes, the backup software told them that it couldn't read them. The only way to ensure that this never happens to you is to run regular verification tests against your media. There are several different types of verification:

Reading part of volume and comparing it

> There is at least one major vendor that works this way. If you turn on media verification, it forwards to the end of the volume and reads a file or two. It compares those files against what it believes should be there. This is obviously the lowest level of verification.

Comparing table of contents to index

> This is a step up from the first type of verification. It is the equivalent of doing a `tar tvf`. It does not verify the contents of the file; it verifies only that the backup software can read the header of the file.

Comparing contents of backup against contents of filesystem

> This type of verification is common in low-end PC backup software. Basically, the backup software looks at its backup of a particular filesystem, then compares its contents against the actual contents of the filesystem. Some software packages that do this will automatically back up any files that are different from what's on the backup or that do not exist on the backup. This type of

verification is very difficult, because most systems are changing constantly.

Comparing checksum to index

Some backup software products record a checksum for each file that they back up. They then are able to read the backup volume and compare the checksum of the file on the volume with the checksum that is recorded in the index for that file. This makes sure that the file on the backup volume will be readable when the time comes.

Verify, Verify, Verify!

We were using commercial backup software to back up our file servers and database servers. One day, a multimillion-dollar client wanted some files back that were archived about a year and a half before. We got the tapes and tried a restore. Nothing. We tried other tapes. Nothing. The system administrator and her manager were both fired. The company lost the client and got sued. The root cause was never identified, but they had definitely never tried to verify their backups.

Eugene Lee

Cost

The pricing aspect of backup software is too complex to cover in detail here, but suffice it to say that there are a number of factors that may be included in the total price, depending on which vendor you buy from:

- The number of clients that you want to back up
- The number of backup drives you wish to use
- What type of backup drives you want to use (high-speed devices often cost more)
- The number of libraries and the number of drives and slots that they have
- The size of the systems (in CPU power)
- The speed of backup that you need
- The number of database servers you have
- The number of different types of databases that you have
- The number of other special-treatment clients (e.g., MVS, Back Office) that require special interfaces
- The type of support you expect (24/7, 8/5, etc.)

Vendor

There is a lot of important information you need to know about a company from which you plan to purchase such a mission-critical product as backup software. How long has it been providing backup solutions? What kinds of resources are dedicated to the products' development? What type of support does it have? Is is open to suggestions about its product?

Get the name of at least one reference site, and talk to them. Be aware, it is very hard for companies to come up with references. A lot of clients do not want to be a reference, just for political and legal reasons. Be flexible here. Don't require the salesperson to come up with a reference site that is exactly like your environment.

> **Tip ✎**
>
> If you do get a reference site, make sure that you get in touch with them. The number one complaint of salespeople is that they go through the trouble of obtaining reference sites, only to have the customer never call them.

The Internet is also a wonderful asset at a time like this. Search for the product's name in the Usenet archives at http://www.google.com/groups. Look for names of people who say good and bad things, and then email them. Also, do general searches on all search engines. A really good product will have references on consultants' web sites. A really bad product might even have a http://www.backupproductsucks.com web site. Read everything with a grain of salt, and recognize that every single vendor of every single product has a group of people

somewhere who hate it and chose someone else’s product. Some clients have been through three backup products and are looking for a fourth.

Final Thoughts

Picking a commercial backup utility is a hard job. The only job that's harder is writing one. Take the time to do it right, and don't be afraid to get professional help in making this decision; it's a constantly moving target.

Also, try not to dismiss too quickly features you believe you will not need. I would submit that you might need them at some point. You never know how your environment will grow. For example, my first commercial backup software setup was designed to handle around 20 machines with a total of 200 GB. Within two years, that became 250 machines and several terabytes, and we picked up several different operating systems and database types along the way. You just never know how big or complicated your environment is going to grow. However, if you are simply looking for a backup solution for a small environment, and you are sure that it will never grow much larger, feel free to ignore questions about features for very large environments. (If you are a small environment, perhaps you are better served by one of the open-source tools covered earlier in the book.)

I wish you the best of success in finding a product to suit your needs.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 9. Backup Hardware

How do you decide which type of backup drive to purchase? There are big drives, small drives, fast drives, slow drives, quick-acting drives, slow-acting drives, tape drives, optical drives, disk drives behaving as disk drives, disk drives emulating tape drives and tape libraries, and disk targets with data de-duplication features. This decision actually is easier to make than you might think. It's certainly easier than deciding on a backup product. There are only eight critical decision factors in the decision, and this chapter covers each of these factors in detail—including data to help you decide which backup drive is appropriate for you. Specific manufacturers' offerings are presented briefly, but due to the changeable nature of this information, they will not be covered in detail in this book. This chapter also discusses common questions about backup hardware, such as compression, density, cleaning, and media usage. Hopefully it will answer your questions in this area and assist you in getting the most out of your backup hardware.

Tip ✎

As in other places in the book, please note the use of the term *backup drive* here. More than ever before, it's a good possibility that your backup drive may not be a tape drive. It may be a disk drive acting like a disk drive, a disk drive emulating a tape drive or tape library, or some type of optical drive.

Decision Factors

As mentioned previously, there are only eight decision factors to consider when deciding which backup drive to buy: reliability, duty cycle, transfer speed, flexibility, time-to-data, capacity, removability, and cost. You need to look at each of the eight decision factors and decide which are most important to you and your data. For example, an environment with a 6 TB database might care about reliability first, transfer speed second, and cost last. On the other hand, an environment that is going to use some type of Hierarchical Storage Management (HSM) system will be concerned about reliability first, time-to-data second, and transfer speed last. Let's take a look at these eight decision factors now.

Reliability

Any electronic repair shop will tell you that moving parts fail. If a mechanism to be repaired contains 25 circuit boards and 1 spinning wheel, they always check the spinning wheel first. The bad news is that backup drives—and especially tape drives—have more moving parts than any other part of your computer system. This means that, statistically speaking, a backup drive has a much greater chance of mechanical failure than a CPU.

> **Warning ☠**
>
> Cheap tapes can cause data loss. Don't buy cheap tapes!

Drives that fail on a regular basis affect your system availability because replacing some drives requires shutting down the system. Unfortunately, what usually happens is that the replacement of malfunctioning drives is very low on the priority list. A bad drive may wait several

days or even weeks to be replaced, even if the replacement part is available immediately. (This is because swapping drives sometimes requires system down time.) So, if drives fail too often, their failure can significantly affect the overall integrity of the backup system. That is, drives could fail so often that a large backup or restore would not have enough drives available to complete within a reasonable timeframe. This is why drive reliability is a very important factor in deciding what backup drive to get.

Drive manufacturers use the *mean-time-between-failure* (MTBF) value to represent the reliability of their drives. Unfortunately, these numbers usually are derived from artificial environments that attempt to simulate thousands of hours of use. (How else would a drive that has been on the market for only six months be able to advertise a 30,000-hour MTBF? That's almost 3.5 years.) The best source of information about the reliability of different drives is the Internet. Use several search engines to locate discussions about the backup drive in question. Google is definitely helpful here. Search its complete archive for different phrases that might locate discussions about the drive you are considering (e.g., DLT, LTO-4, T10000, and so on). If you don't find any, post your own message and see who replies. (The most useful Usenet group for this topic is *comp.arch.storage*. You do remember Usenet, right?) After that, you should do the unthinkable—actually talk to people. Before you buy a new type of backup drive, I would talk to no less than five companies using that drive.

Desperate Times

I have been saved a number of times with the "twist-start" recovery method. You know, when you get a hard drive that doesn't seem to be spinning up right, and you give the drive a quick twist of the wrist. Magically it gets up to speed and works for a little while.

Brian O’Neill

While the removability of the media is an important feature of tape drives, it requires the drives to be an open system. Dust and other contaminants are introduced in the drive every time the door is opened to insert or eject a tape, and contaminants are the death of any mechanical system. In contrast, disk drives are closed systems; the media is never separated from the drive, and no air is ever allowed to enter the unit. This is a major reason why disk drives are inherently more reliable than tape drives, which is why they are becoming increasingly popular as backup drives.

Finally, another reason why disk drives are more reliable than tape media is that you can use RAID to mitigate the risk of a single piece of media causing you real damage. With tape or optical, you make a backup, set it on the shelf, and hope the bits don’t fall off or get rearranged while it’s sitting there. You never really know, it’s any good until you use it for a restore. With RAID-protected disk, you can constantly monitor the health of a piece of media. If any disk drives start to look unhealthy (or fail altogether), just replace them, and have the RAID system rebuild them. No lost data, no failed backups—just a little maintenance.

Duty Cycle

When comparing the MTBF of different drives, don’t forget to look at the *duty cycle* the drive was designed for. A drive’s duty cycle measures the percentage of time a backup drive spends reading, writing, and verifying data. For example, if a particular drive operates for 6 hours a day, that drive has a 25 percent duty cycle. If it operated continuously, it has a 100 percent duty cycle.

Each drive has an intended duty cycle. Fibre Channel and SCSI disk drives, for example, are usually designed for applications demanding a 100 percent duty cycle; ATA disk drives are designed for lower duty cycles. This is why Fibre Channel and SCSI disk drives are often used in high-volume applications, such as database servers or very busy file servers. ATA disk drives have historically been used in PCs and laptops where the demands on the disk drives are much lower. Even now that they're being used in large storage arrays and virtual tape libraries, these storage systems are usually intended for reference data. Backup, archive, and imaging systems create reference data. You make backups every day, but any given backup is rarely accessed—if at all. The same is true of imaging and archive systems. If your company creates an image of every customer contract, for example, any given contract is stored once and only occasionally (if ever) accessed.

Tape drives also have intended duty cycles. Mainframe-style tape drives are typically designed with a 100 percent duty cycle whereas those intended for open-systems markets are designed for lower duty cycles. This is because mainframes typically use their tape subsystems as a secondary storage system while open systems typically use tape drives only for backup. Mainframes are constantly reading and writing production data to and from tape. Therefore, their tape drives must be prepared to operate continuously, and mainframe tape libraries are judged on how many exchanges they can perform per hour. In contrast, most open-systems backup applications use their tape drives for less than half the day, which is why tape drives designed for open systems typically have much lower duty cycles.

It's important to buy a tape drive with a duty cycle that fits your application. Not doing so can significantly increase the cost or decrease the reliability of your backup drive. If

you buy a drive with a 40 percent duty cycle and use it continuously, just expect that drive to fail much sooner than its published MTBF. If you have an application with a 40 percent duty cycle, and you buy a drive that was intended for a 100 percent duty cycle, expect to pay as much as 300 percent more for the same speed and capacity—sometimes even less speed and capacity! The vendor with the drive that has a 100 percent duty cycle may claim their drive is more reliable, and they're right. If that drive was used in a 100 percent duty cycle environment, it will be more reliable than the drive with a 40 percent duty cycle used in that same environment. However, a tape drive designed for a 40 percent duty cycle should be just as reliable as the 100 percent duty cycle drive if it is used in a 40 percent duty cycle environment, such as traditional backup.

Transfer Speed

Transfer speed is also very important to consider when deciding on a backup drive. How fast can the drive read and write data? When comparing different drives, of course, you must compare *native transfer speeds*. A drive's native transfer speed is its speed without compression. This often is difficult to assess, because many drives quote only the compressed numbers. They usually attach a footnote to that number that says something like, "This number assumes a 2-to-1 compression ratio. The actual transfer speed varies based on the compressibility of the data." The reason for this is that different manufacturers make different claims regarding compression ratios. Some vendors have claimed a typical compression ratio of as much as 5 to 1. While it is true that different compression algorithms do yield different results, it is very difficult to verify a particular vendor's claims of better compression.

To make sure that you are comparing apples to apples, always compare native transfer speeds.

Some vendors do not use the term *native* transfer rate. They might use the term *head-to-tape* transfer rate, which refers to how fast the recording head can write the data on the tape. This rate does not change with compression. If the data is compressed prior to being sent to the recording head, the drive's effective throughput is increased, but the head-to-tape speed does not change.

When looking at drive specifications, try to locate the word *sustained*. Sustained transfer rates provide the fairest comparison between drives. Some drives quote *burst* rates and *synchronous* rates, which are both temporary, best-case scenarios. (Based on your application, you may wish to compare burst and synchronous transfer rates as well; just make sure you know that's what you're doing.)

Don't forget to put everything in the same terms. Some drives publish their numbers in gigabytes or megabytes per hour. To get megabytes per second from megabytes per hour, just divide the number by 3,600, which is the number of seconds in an hour. If it's gigabytes per hour, you can get megabytes per second by dividing the number by 3,686,400, which is 3,600 times 1,024.

Flexibility

A backup drive can be flexible in two different ways: it can respond well to different data rates, and it can be used in different ways. Generally what you find is that tape and optical drives are not very flexible, and that disk drives are fairly flexible. Let's explore this idea.

Tape drives: Not so flexible

Tape drives do not generally respond well to different data rates, they can be used for only one purpose, and you generally need at least one per backup server.

Tape drives must be streamed. That is, they generally need to be sent a stream of data at close to their maximum throughput rate in order to consistently operate well. If they receive a stream of data slower than that, they actually can write slower than the incoming data stream and can also fail more often. See the “Tape Drives” section later in this chapter for more details on this concept.

It may seem silly to say it, but tape drives can really only do two things. They can write data sequentially and read data sequentially. This means that they can be used only with applications that understand how to read and write to tape drives—in other words, backup and archive applications.

In addition, if you’ve got five backup jobs, you need five tape drives. Yes, many commercial backup software products can interleave onto a single drive jobs that are sent to a single backup server. However, they cannot interleave jobs when they’re sent to multiple backup servers. Multiple backup servers often can “time-share” drives, as long as they make sure that no two servers write to the same drive at the same time. However, they cannot actually read and write to the same tape drive simultaneously.

Optical drives: A little more flexible

Optical drives are more flexible than tape drives, but aren’t quite as flexible as disk drives. They can easily handle different incoming data rates, but they still tend to have the single-minded issue that tape drives have.

Optical drives are random-access devices, which means that they don’t have the issues that tape drives have with

streaming. Generally, they can handle any data rate up to their native throughput rate. As of this writing, no optical drives use hardware compression, so native throughput rate is the same as effective throughput rate for optical drives.

Since they behave more like disk drives than tape drives, optical drives can also usually be mounted as a filesystem, which means that they also don't have the single-minded issue that tape drives have. Multiple applications can read and write simultaneously to the same optical drive, just like they can to any mounted filesystem. Global filesystem software could also be used with an optical drive to allow multiple servers to simultaneously read and write to it as well.

Optical drives could be made to be as flexible as disk. For example, someone could write drivers that would make an optical drive look like a tape drive. As of this writing, no one is doing that.

Disk drives: Very flexible

Disk drives really win in the flexibility category. They can handle any data rate you send them, even going beyond their native throughput, through the use of RAID. They can emulate just about any kind of backup device you want, and they can emulate as many of those devices as you would like.

In addition to being able to handle slower data rates, disk drives can be striped together using RAID to handle very large incoming data rates. A single RAID array can handle hundreds of megabytes per second, or it can handle 1 Kbps —whatever you need.

Thanks to virtualization drivers, disks can also emulate other things. RAID is really several disks emulating one big

disk. Virtual tape software allows disks to emulate tape. With virtual tape software, you see a disk array, but your backup server sees a tape drive or a tape library. This is referred to as a *virtual tape library*, or VTL.

Not only can disk drives emulate tape drives, they can emulate a lot of tape drives. Virtual tape software allows a single disk array to emulate as many tape drives or tape libraries as you like, allowing several servers to easily share the same resource without using anything like global filesystem software. Unlike real tape drives, you can make more virtual tape drives with the push of a button.

There's another interesting way that disk drives are flexible. If your backup software supports multiplexing, and you choose to use that feature when backing up to a virtual tape drive, they can help mitigate the side effects of multiplexing by going significantly faster than a normal tape drive.

> **Tip ✎**
>
> Normally, you would not multiplex to a virtual tape drive. The reason you multiplexed to tape was to stream the tape; this is not necessary with a virtual tape drive. However, there are some licensing issues with some VTLs and some backup software packages that may give you reason to multiplex to a VTL. The point of this section is to illustrate that if you chose to multiplex to a virtual tape, it might not impact your restore performance as much as multiplexing to physical tape.

Here's how it works. Suppose your backup software multiplexed several backups to a single tape, and you ask it to restore just one of those backups. What it actually does

is read all the multiplexed backups and throw into the bit bucket all the bits it doesn't need. If your tape drive can go only 50 MBps, it means that a significant portion (75 percent with a multiplexing setting of four) of that 50 MBps is being spent reading data that's being thrown away. However, what if you had a "tape drive" that could read data at 150 MBps? If you threw away 75 percent of that, you'd still have 37.5 MBps, which is a respectable transfer rate.

Finally, the truth is that the virtual tape interface is just that—an interface to the disk drive. In the end, the data is still lying on disk, which leaves room for a number of possibilities that are covered in more detail later in the chapter. They include de-duplication, replication, content-awareness, and re-presentation.

Time-to-Data

Transfer speed is not always the most important deciding factor. (In fact, it's starting to be one of the last things I personally look at.) Obviously, having a faster drive makes backing up and restoring large amounts of data easier. However, many restore requests are for a single file or a small group of files. Depending on your environment, this may account for 99 percent of your restores. In addition, many large restores require anything from several tape mounts to thousands of them. (I know of one restore that required mounting 20,000 tapes.) What counts most in these restores is *time-to-data*: how long does it take to load a volume, seek to the appropriate place on the volume, and start reading data? Streaming tape drives usually are "slower on the draw" and therefore have a much longer time-to-data. The winners in the time-to-data category used to be optical drives. The worst time-to-data value for an optical drive is typically around 12 seconds, allowing 5

seconds for a robotic exchange. If the file being restored is on a platter that is already loaded, time-to-data is less than a second. However, the new winner in this category is the disk drive. With access times in the nanoseconds, it's hard to beat a disk drive—even if the disk drive is emulating a tape drive. How long does it take to rewind, eject, or load a virtual tape?

HSM applications place the highest importance on the time-to-data value because of how HSM works. An unused file is automatically migrated from disk to a less expensive storage medium. The user does not realize this and may eventually attempt to access the file. The HSM system detects that the file is needed and automatically retrieves it from the backup medium. All the user sees is a delay in accessing the file. If the delay is 12 seconds or less, the user may not even notice. However, suppose that the file is placed on a tape drive whose time-to-data is two minutes. A typical user is either going to call the help desk or reboot by that time. That is why HSM environments should use either optical media or one of the newer tape drives that have been designed with very low time-to-data values.

In extremely high-volume HSM environments, it also matters how long it takes to rewind and eject the volume. The time-to-data is added to the data-transfer time and rewind-and-eject time to create something called *cycle time*. If the cycle time of a particular tape drive is 2 minutes, the HSM system can service 30 file migration requests per hour per drive. An 8-drive autoloader would increase that to 240 files per hour. In comparison, an average optical platter has a cycle time of less than 30 seconds. This means an 8-drive autoloader using optical platters can service up to 1,000 migration requests per hour. Consider carefully how important the time-to-data value is in your environment.

Capacity

Capacity used to be the most important factor when considering a backup drive. It still is in many nonautomated environments. If a full backup of your entire system can fit on one volume, you do not have to worry about swapping volumes in the middle of the night. However, using an autoloader significantly reduces the importance of volume capacity. In fact, having your data on multiple smaller volumes may actually make a restore go faster with today's backup software.

Having a drive of insufficient capacity affects your overall transfer rate, though. Suppose that you are backing up to a 100 GB volume with a transfer rate of 100 MBps and a cycle time of 4 minutes. At 100 MBps, you can fill the entire volume in a little over 16 minutes. You would need to swap volumes, which would take 4 minutes. That means that it actually took 20 minutes to back up the 100 GB, reducing your overall effective throughput to 80 MBps—not 100. Suppose that there was a bank of 8 of these drives in an autoloader; the overall throughput for an 8-hour period would be reduced from 29 TB to 23 TB—a 6 TB difference. If your volume had a capacity of 500 GB, though, the amount of transferred data lost due to volume swaps is reduced to around 1 TB.

Also remember that larger capacity usually means a longer time-to-data—especially in tape drives. (Obviously, it takes longer to get halfway through a 2,000-foot tape than it does to get halfway through a 1,000-foot tape.) Some tape drives mitigate this by mounting midpoint, but it's still a long way to the end of the tape—even if you start in the middle.

Another capacity consideration is the intended use of the drive. Are you planning to use backup software that writes continuously to each volume until it is full or will you make

lots of small backups to a single volume? For example, suppose that you regularly make database exports to tape and send them to a partner who imports that data into his database. Do you really need a 400 GB tape to export a 500 MB database? Every export will require an expensive 400 GB tape. It would be more cost effective to use a cheaper, smaller tape or optical drive for such a purpose.

Removability

With the advent of virtual tape, we need to consider this aspect of backup drives. Obviously, tape drives and optical drives are removable, and it is considered one of their great features. As of this writing, removable virtual tape cartridges are also beginning to be available. However, most virtual tape and disk units do not have removable media available.

Many customers are beginning to ask whether or not removability is still a requirement for them. The reason that most people need removability is that they want to be able to send their tapes to an off-site vaulting vendor. What if you could do that without moving tape around? Would you still need the media to be removable? Ask yourself this question when examining the replication features of disk targets, covered later in this chapter.

Cost

Drives that are more reliable and flexible, store more data at a faster rate, and have a smaller time-to-data value usually cost more. The only time when you may get a price break on a reliable, quick drive is if a manufacturer is trying to break into a market that is dominated by another manufacturer. Of course, then you bear the risk of the drive

being taken off the market. (There have been a number of dead soldiers over the years.)

Another cost factor to consider is reusability of media. There are several *write-once-read-many* (WORM) technologies that do not allow you to reuse media. These drives have a definite purpose, such as making bootable CDs or unchangeable archives, but you definitely should consider whether you are allowed to reuse the media when calculating total cost of ownership.

Another thing you should consider is the cost involved in managing the intelligence behind the backup system. If you use tape drives, you need to do a lot more design and management of the system to make sure that the system is writing data to tape in a way that utilizes tape drives well. If you use disk, you hardly have to do that, thus significantly reducing your total cost of ownership.

Make sure you consider all cost factors when comparing the cost of the different media types. This is especially true when considering the cost of your on-site storage system because it probably won't be a standalone tape drive. It most likely will be either a tape library, an optical library, or some kind of disk system. Do not make the mistake of comparing only the price of the media: tape will always win. Remember that when it comes to your on-site storage system, that piece of media is worthless without the tape or optical library it goes in. Start with determining how much storage you need based on how many full and incremental backups you're going to store on-site. Suppose, for example, you have determined that you need 200 TB of on-site storage. Compare, then, the cost of a fully populated 200 TB tape library, a fully populated 200 TB optical library, and a 200 TB disk system. A fully populated tape or optical library has all the tape/optical drives you need, all the slots you need, and enough tapes/platters to fill those slots. A

fully populated disk system has all the disks you need, along with any RAID or virtualization software or hardware needed to make it behave the way you want it to behave. (Of course, you have to take into consideration the differences in power requirements between a disk system and a tape system.) When you compare things like this, you might be surprised at the numbers, especially when you factor in the data de-duplication features of disk systems.

Summary

While several drives fit into the middle-of-the-road category, most drives excel in one or more of these eight critical factors. Tape drives have historically excelled in the cost of acquisition area; however, the de-duplication features of disk units are changing that. Different backup drives are better than others in the areas of time-to-data, transfer rate, and flexibility. Optical media obviously provides a quick time-to-data, but they definitely have smaller capacity yet cost more than their tape counterparts. Disk drives *can* be more expensive than either tape or optical media, but not necessarily. They excel in every other decision factor. You must decide which factors are most important in your decision.

Using Backup Hardware

People ask a number of questions about using backup hardware. Hopefully this section will help to answer them.

Compression

To save space, data can be compressed before being written to the drive. There are two methods of compression: software and hardware. Software compression is performed by compressing the data via software prior to sending it to the drive. When using hardware compression, uncompressed data is sent to the drive, and a specialized chip on the drive does the compression. Figure 9-1 tries to show the paths that the two different types of compression take.

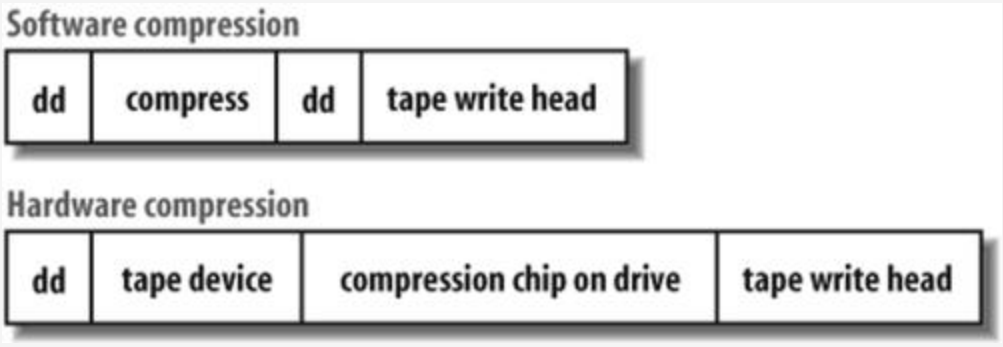


Figure 9-1. Data paths of software and hardware compression

Software compression obviously requires more usage of the host CPU than does hardware compression. This CPU usage usually outweighs its one advantage—that it lowers the amount of data transferred across the network. Hardware compression, on the other hand, actually makes things go faster. The specialized compression chip can compress data at line speed. Since it's reducing the amount of data actually being written to tape, and it's doing so at line speed, it actually increases the effective throughput of

your drive. Therefore, if you have a tape drive that can write at 120 MBps, and the incoming data allows for a 2-to-1 compression ratio, the drive can accept data at 240 MBps. It compresses the incoming 240 MBps stream and generates a 120 MBps output that is then written to the actual tape drive. If the incoming data can be compressed 3 to 1, the drive can accept data at 360 MBps and generate the same 120 MBps output for the tape drive.

The amount of compression that you achieve is highly dependent on the compression ratio of the data. Certain kinds of data compress better than others. Text data, for example, compresses very well. Certain kinds of image formats (e.g., TIFF) are already compressed and cannot be compressed further. If you are backing up a filesystem containing nothing but TIFF files, you'll probably achieve no more than the native speed of the drive. However, if you are backing up the filesystem containing nothing but logfiles, you may get transfer rates that are well in excess of the published rates of the drive, because most of them are based on a 2-to-1 compression ratio.

Density Versus Compression

Density and compression often are confused. As I mentioned earlier, hardware compression actually uses compression algorithms to compress the data prior to sending it to the actual recording head. The way that the recording head writes the data to the drive actually doesn't change. Density, on the other hand, refers to how densely the data is physically written on the tape. Density is represented by a bits per inch, or bpi, value. This is how later generations of tape drives achieve higher throughput and capacity even with the same physical tape. If a drive moves the tape at the same speed but can write bits closer

together on the tape, the drive can write the data faster and can fit more on the tape.

How Often Should I Change My Media?

Media life is described by manufacturers in terms of number of passes. A pass is any time that the medium passes by the recording head; this means every time a tape is written to or read from, as well as any time the tape is retensioned. Most manufacturers of data-grade media specify that a given piece of media can survive several thousand passes. If you were streaming the tape drive and performing one backup and one restore per week, it would take almost 20 years to reach 2,000 passes. It's important to realize the importance that shoe-shining plays in this area. If a tape drive is *shoe-shining*, each time it moves a section of the tape back and forth across the head counts as a pass. Therefore, if a drive is shoe-shining a lot, one backup can result in 20 passes over each section of the media, reducing the media life from 20 years to 1 year.

How many times you reuse a piece of media should be based on your environment. I could not in good conscience recommend that you throw away media after only a few uses. I also could not recommend that you never replace your media. I believe that you should adopt a wait-and-see approach to replacing media. Perform regular backup and restore tests on your media. Pull volumes out at random and attempt to restore files from them. Replace a volume if it gives the slightest indication of trouble. If you see a trend of a particular bad batch of tapes, you should perhaps do more testing to see if the problem is widespread. If you watch your backup logs, you also may occasionally see I/O errors when writing to a particular tape. That tape also should be replaced.

Cartridge Care

Today's backup media is much more resilient than it used to be. (I remember accidentally dropping a nine-track tape on the floor and watching it unravel all over the place.) You still need to treat media with care, though. Tapes should be stored with the spline up and the axis of the tape spool parallel to the ground. (If this were a cassette tape, and you stuck a pencil through the tension wheel, the pencil would be pointing straight out—not straight up. You do know what a cassette tape is, right?) This prevents gravity from causing the tape to settle around the spool.

If you accidentally drop a tape, pick it up and shake it. If you hear noise, do not use the tape. There are delicate mechanisms inside that allow the tape to be brought back into the cartridge. Dropping a tape can cause one of the springs to pop off its pedestal. If you then put that tape into a drive, the tape will be spooled into the drive, and you will not be able to pull it out. The only way to repair it is to disassemble the drive.

Drive Care

Clean your drive according to the manufacturer's recommendations. So many problems can be prevented by this simple maintenance step. Read the manual for your particular model of drive and follow the directions that you read there. What else can I say? Also remember that many manufacturers are extremely hard to deal with when repairs come up if you have not religiously followed the maintenance schedule.

Tip ✎

Please note that some drive manufacturers want you to clean their drives only when the drives request to be cleaned. These drives are built to be self-cleaning and should get dirty only under adverse circumstances. These drives are made to notify you when they need to be cleaned, and cleaning them more often than they request to be cleaned can actually shorten the life of the drive.

Nearline and Offline Storage

Two terms that you may see when considering the purchase of backup hardware are *nearline* and *offline*. Offline is what you typically think of when you're thinking about backups. It is a second copy of online data. It is not intended to be the primary copy of the information, unless of course it is needed for a restore. A nearline copy, though, is completely different. "Nearline" implies that a file is "close to being online." That is, it is stored in an automated library on a less expensive storage device. It may take several seconds, or even a minute, to get the data, but it can be automatically retrieved. An HSM implementation requires nearline storage.

Tape Drives

Here’s some helpful information about tape drives.

Tape Drives Must Be Streamed

Modern tape drives do not generally respond well to different data rates. That is, if a tape drive was designed to read or write data at 120 MBps, then it will *typically* perform well only if it’s reading or writing at 120 MBps, with some exceptions that are covered later in this chapter in the section “Variable Speed Tape Drives.” Most tape drives today can actually only write data at their rated speed. If you send it 120 MBps, it will write at 120 MBps. However, if you send that same drive at 60 MBps, it will spend 50 percent of its time writing at 120 MBps and the other 50 percent of its time *preparing* to write at 120 MBps.

Here’s how it works. You start sending the data to the drive, where it first goes to the drive’s buffer. The tape drive starts spinning up its mechanics, preparing to write the data from the buffer to the tape at 120 MBps (if it’s a 120 MBps drive). Once the buffer is full, the tape drive starts reading data from the buffer and writing it to the tape. If you’re streaming the drive, you’re filling the buffer with new data as fast as the drive is emptying the buffer as it writes data to tape.

However, if you’re not filling the buffer as fast as it’s being emptied, at some point, the tape drive looks at the buffer and it’s not full. The tape is still moving, of course, because it needs to be moving to write data—but now there’s no more data to write. The tape drive must stop, rewind the tape, move it back to a position before it stopped writing, and then wait for the buffer to fill up with data again. This

is called *repositioning*, and every reposition takes a finite amount of time, sometimes as much as a few seconds. When the buffer is full again, the drive starts moving the tape again, and the buffer gets emptied again.

If the buffer is constantly full of data, and the tape drive never has to reposition, we say that the tape drive is *streaming*. If you are sending anything less than the drive's target throughput rate, the drive is repositioning. If you're sending a stream of data that's less than, but close to the target throughput rate, it's repositioning occasionally. The slower the data rate is in comparison to the target rate, the more the drive is repositioning. If the drive is repositioning a lot, the tape is being constantly moved back and forth across the read/write head like a shoe-shine rag across a dress shoe, which is why this scenario is called *shoe-shining*.

It usually takes longer to reposition the tape than it does to empty the buffer. Therefore, once a reposition has started, it doesn't matter how quickly you can fill up the buffer; the drive has to finish its reposition before it can start writing and empty the buffer again. Meanwhile, the buffer might be full, but it isn't being emptied. This means that the incoming data must be told to wait while the drive prepares to empty the buffer. The more often this happens, the less the drive is able to keep up with the incoming data rate. This is why a tape drive that is not streaming may actually write data at a slower rate than the incoming data rate; the drive is spending all of its time repositioning, and very little of its time writing data.

Compression Makes It Harder to Stream Drives

Although you should not use compression rates when comparing different backup drives, it is important to

understand the role that compression plays in determining actual throughput. As mentioned in the section on compression earlier in this chapter, a lot of people do not realize that compression increases the effective throughput rate of your drive by the same ratio it increases a drive's effective storage capacity. In the previous section, I mentioned that a 120 MBps tape drive should be sent data at the rate of 120 MBps if you expect it to stream. This is called its *target throughput rate*. However, if the data going to a 120 MBps drive is being compressed 2 to 1, its effective target throughput rate is actually 240 MBps. A drive is streaming only if it's writing data at or near its effective target throughput rate.

You need to determine the effective target throughput rate of the drives in your environment. To do this, you must determine what compression ratio you're actually getting, then multiply that ratio times your drive's target throughput rate (e.g., 120 MBps) to get your effective throughput rate (e.g., 240 MBps). To determine your compression ratio, you need to determine how much data is being written to your full tapes. Most backup products write data to a tape until they hit the PEOT mark. This means that if you divide the average size of a full tape by the native capacity of a tape, you will end up with your actual average compression ratio. For example, if the native capacity of your tapes is 400 GB, and you're fitting an average of 600 GB of data on each full tape, you're getting an average compression ratio of 1.5 to 1. Therefore, your 120 MBps tape drive is actually a 180 MBps tape drive.

Variable Speed Tape Drives

The previous section explained how tape drives do not handle varying data rates very well because they can write

at only one speed. As mentioned previously, this is really a function of the need to have the recording head move across the media very quickly to obtain a high signal-to-noise ratio.

Having said that, tape drive vendors had gotten tired of hearing about this old complaint, and they were genuinely worried that the problem was going to get worse and worse as tape drive speeds got faster and faster. The faster drives got, the harder it would be to stream them, and the better the chance they would shoe-shine. Yet the nature of the market required them to come out with faster and faster tape drives.

So a few bright vendors started to realize that if they came out with a faster drive that could also go slow, they'd have something truly special. Therefore, some drives are now able to step down their speeds to keep up with slower data rates. As of this writing, there are drives that can go as slow as $^{1}/_{2}$ of the original native transfer rate; in other words, a 100 MBps drive that can also run at 50 MBps without shoe-shining.

> **Warning ☠**
>
> Some vendors are claiming that variable speed tape drives make shoe-shining a thing of the past. This is absolutely not true.

Variable speed tape drives can still suffer from shoe-shining. Suppose a tape drive has a native speed of 120 MBps, and your data is compressing at 1.5 to 1, turning that drive into a 180 MBps tape drive. A variable speed tape drive can typically slow down to 50 percent of its original speed, allowing the drive in our example to become a 90 MBps tape drive after compression. If you are

supplying data slower than 90 MBps, then it is not streaming; *it is shoe-shining*.

Helical and Linear Tape Drives Are Different

It's generally held that helical scan tape drives suffer less than linear tape drives when it comes to slow incoming data rates. A review of the differences between these technologies will help explain why.

One of the best ways to illustrate the difference between helical scan and linear recording technologies is to look at a non-hi-fi VCR, because it actually incorporates both technologies and illustrates an important point. Do you remember VCRs before they all went hi-fi? Did you ever record and watch a movie on a non-hi-fi VCR using the extended play (EP) setting? When you played that tape, it sounded horrible.

> **Tip ✎**
>
> This analogy worked much better 5–10 years ago when not all VCRs were hi-fi. Some readers may have to go ask their parents what a non-hi-fi VCR was. If you've never had a non-hi-fi VCR, you'll just have to trust me. Recording a movie on EP sounded like garbage!

Yet if you record the same movie on the same setting with a hi-fi VCR, the audio sounds fine. Have you ever wondered why?

Look at Figure 9-2. A VCR's tape is brought out of the cartridge and wrapped around a rotating drum. As you can see in Figure 9-3, the drum is angled slightly and has recording heads on its side. (The rectangle sitting at an angle in Figure 9-3 represents the angled drum with its

rotating recording heads.) As the tape is pulled slowly around the drum, the diagonally positioned recording heads write “stripes” of video data diagonally across the tape, as can be seen in the bottom of Figure 9-3. Although the tape is moving very slowly around the drum, the drum is spinning very fast. This means that the recording heads on the edge of the drum actually are moving across the tape very quickly, resulting in a good quality video signal.

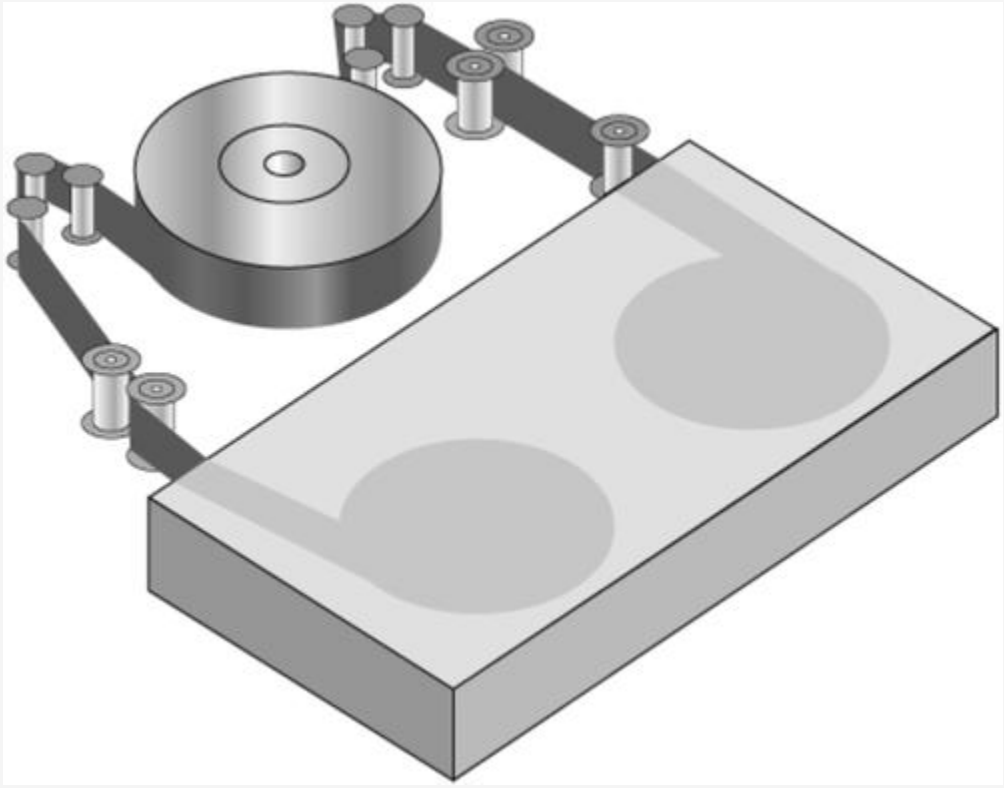


Figure 9-2. VCR tape path

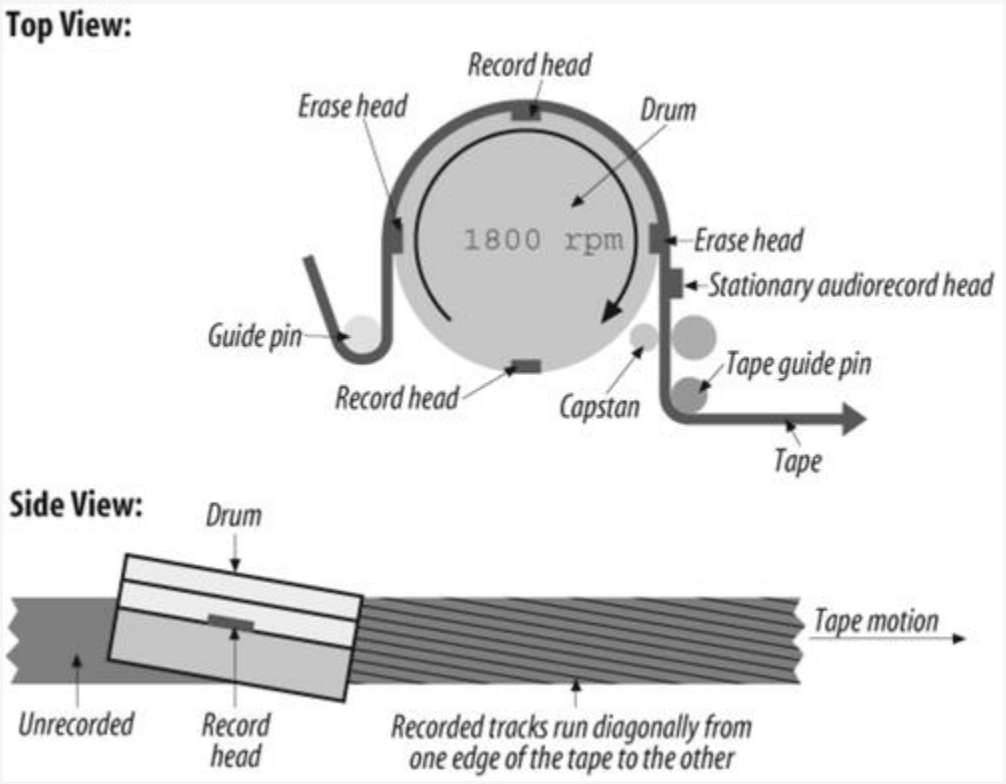


Figure 9-3. Helical scan recording

The drum spins at 1800 rpm, or 30 revolutions per second, with one head on each side of the drum. This means that the recording heads are writing a stripe of data 60 times each second. Each one of these stripes contains half of an interlaced video frame. A synchronization signal also is written along the edge of the tape that keeps the tape in sync with the spinning recording heads. The VCR interlaces these images into what you see as full-motion video.

In a non-hi-fi VCR, the tape also passes by a *stationary* audio head that records a linear audio signal along the edge of the tape; this is very similar to how an audiocassette player works. You can see the stationary audio recording head in Figure 9-3. A hi-fi VCR has this same stationary head for backward-compatibility reasons, but it also has audio heads on the spinning drum. This means that it records audio tracks as diagonal stripes alongside the video data, as can be seen in Figure 9-4. The audio recording heads in a hi-fi

VCR are moving across the tape at a high speed, whether you are recording in EP or standard play (SP) mode. This is why a hi-fi VCR can record a good audio signal regardless of the speed at which the tape is moving around the drum.

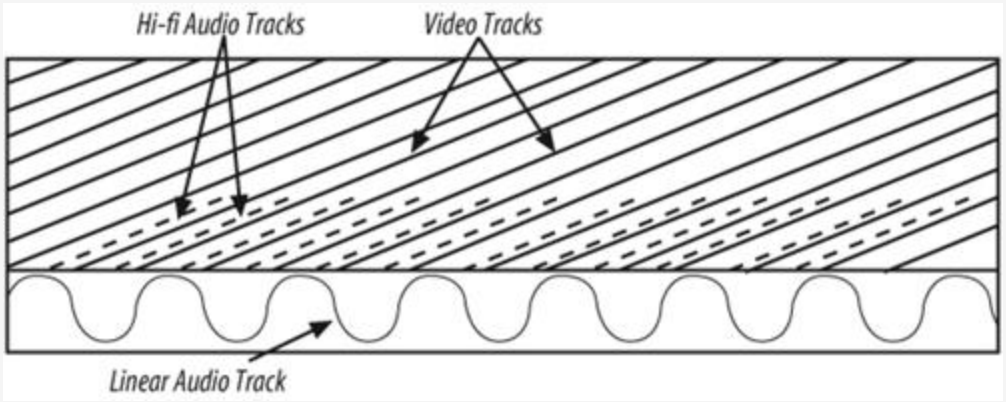


Figure 9-4. A section of videotape

The result is a videotape that looks like the one in Figure 9-4. The video tracks are recorded in diagonal stripes across the tape. In a non-hi-fi VCR, the audio track is recorded slowly, in a linear fashion, along the bottom of the tape as the tape passes by the stationary recording head. A hi-fi VCR also records the audio this way but also records it in quick, diagonal stripes parallel to the video tracks.

Remember that in a non-hi-fi VCR, the video heads move very quickly across the tape, but the audio heads do not. The result is a good-quality video signal, but a poor-quality audio signal. However, when a hi-fi VCR records audio, it records it just like it records the video—in rapid diagonal stripes across the tape. This results in high-quality audio and video signals. This shows how, in order to record a high-quality signal to tape, the recording head must be moved across the media very quickly. This is important because in a data drive, the quality of the signal is everything.

This in-depth explanation of how VCRs work also explains the difference between helical scan and linear recording

technologies. Helical scan drives record data just like a VCR records video, by wrapping the tape around a spinning drum with recording heads attached to it. Linear tape drives move the tape quickly across a stationary recording head. Let us look at these two types of drives in more detail.

As seen in Figure 9-3, a helical drive pulls the tape out of the drive and wraps it around a spinning drum that is turned on a slight angle. On the side of the spinning drum are recording heads that write diagonal stripes across the tape. This allows the tape to move very slowly around the spinning drum, while the recording heads can move across the tape's surface very quickly. This is exactly the same way a VCR records the video signal.

The downside of such a device is that the tape must be wrapped all the way around this spinning drum. Advocates of linear recording technology say that this puts undue stress on the tape. Helical scan manufacturers say that they have changed the way the tape is pulled around the drive in a way that reduces the stress on the tape. Helical scan technology is used in several drives, such as 8 mm, AIT, DAT/DDS, DTF, DTS, and SAIT.

A tape drive that uses linear recording technology pulls the tape very quickly across a fixed recording head. Remember from the lesson about VCRs that the recording head must move very quickly past the tape. Since the recording head is stationary, this requires moving the tape at a very high speed, as much as hundreds of inches per second. One of the popular drives that uses linear recording technology is Linear Tape Open (LTO); its tape path is illustrated in Figure 9-5.

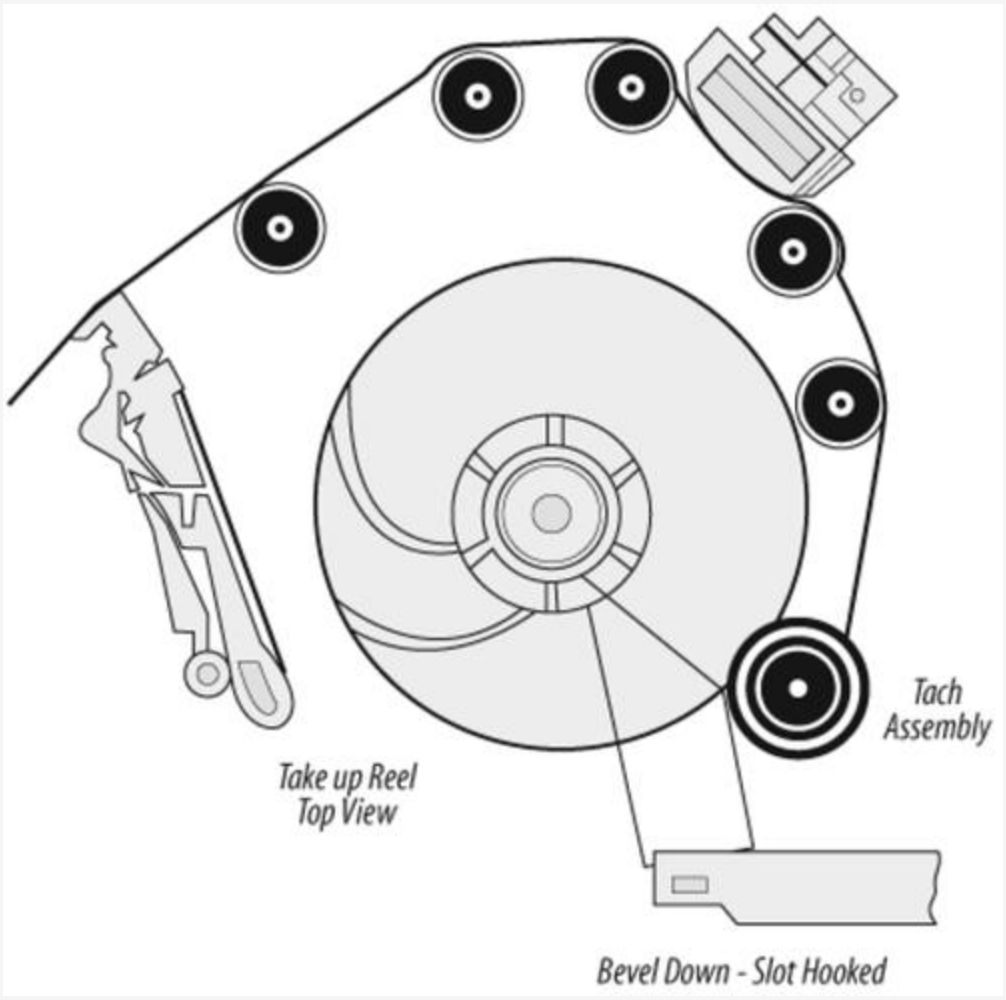


Figure 9-5. Linear recording technology

Most modern linear tape drives, including LTO, use an enhanced linear technology called *linear serpentine*. A drive using the linear serpentine recording method records several stripes of data across the tape from one end to the other. The head then moves slightly up or down and writes another several stripes of data in the reverse direction. Depending on the model of the drive, it may do this several times before it uses up the entire recording surface. Linear technology is also used in a number of tape drives, including DLT, LTO, and all of the drives manufactured by IBM and Sun as of this writing.

Now that you understand the differences between these two technologies, you should be able to see why many feel that helical scan tape drives might suffer shoe-shining less than linear tape drives. A linear tape drive is moving the tape very fast, where a helical scan drive is moving it very slow. A linear tape drive with an empty buffer is going to move past several hundred inches of tape before it "realizes" that the buffer is empty and it begins to reposition. The tape in a helical scan drive, however, isn't moving nearly as fast, so it's won't move that far before it realizes it needs to rewind, and it doesn't have to rewind very far before it gets back to the point it needs to be before it starts writing again. Some time will be associated with syncing up to the sync signal on the bottom of the tape that keeps the heads in sync with the diagonal stripes of data on the tape, but that will probably be faster than what's happening in a linear tape drive.

Cartridges Versus Cassettes

Although many people use the term "cartridge" to refer to any type of tape, cartridges are actually single-spool tapes, such as a DLT, LTO, or SAIT drive. A cassette contains two spools within the tape, such as an AIT or DDS drive.

The reason for discussing this is to explain one primary difference between the way the two types of tape work. A single-spool cartridge does not have a take-up reel inside the cartridge itself. The take-up reel is inside the drive. That means that with single-spool cartridges, the entire tape is pulled out of the cartridge and wrapped around the drive's internal take-up reel.

A cassette's tape, however, effectively remains inside the cartridge. Most technologies pull a certain amount of the tape outside of the cassette at a time, but the bulk of the

tape remains inside the cassette. There are a few technologies that can use a cassette without pulling any of the tape outside of the cassette.

Must've Saved a Bundle

A backup service provider that supported remote backups for several customers scheduled tape pick-ups/drop-offs to occur not at the datacenter, but at a central location. Tapes were transported between the central location and the datacenters in the back of a backup operator's station wagon, sometimes sitting in the back of a hot car for several hours.

Kevin Suttle

Midrange Tape Drive Types

This section briefly covers what the industry tends to call midrange tape drives. A midrange tape drive is typically something that would be considered expensive to a home user but is not a high-end specialized tape drive, such as those found in the black boxes of planes. The tape drives covered in this section currently range from under a thousand dollars to tens of thousands of dollars.

Tip ✎

We do not list the capacity of the drives listed here because they change so frequently. You can easily obtain this information from the Web.

The information I have included is very general and often historical, and is offered mainly to assist you in differentiating among the different types of drives. The drives are listed in alphabetical order as much as possible

so as not to show any preference for any particular drive. Some of the drives covered here are either brand new to the market or not even released as of this writing.

> **Tip ✎**
>
> Some of the drives covered in this chapter have been end-of-lifed by their manufacturers, and I considered dropping them from the book. However, the truly budget-conscious customer just might be buying one of these end-of-lifed drives on the used market. I left them in just for that reader.

3480 (end-of-lifed)

The IBM 3480 drive family has been around awhile; uses a half-inch 3480 cartridge; and includes the 3480, 3490, and 3490E. Although today these drives are made by a number of different manufacturers, they originally were created for IBM mainframes. These drives, therefore, have the longest history of stability and reliability of any drive being sold in the open-systems market today. They are rather slow and small in capacity compared to other tape drives, but they have a relatively quick load and access time.

3590

The IBM 3590 was the successor of the 3480 family, with larger tape capacities and additional speed. It has its own media type—a half-inch 3590 cartridge.

3592

The IBM 3592 was intended to replace the 3590. It uses its own half-inch 3592 cartridge and offers faster transfer

rates and higher capacities than its competitors. There are actually two types of 3592 media. One is less expensive, while the other offers quicker access to data.

TS1120

The IBM TS1120 uses 3592 media and offers extremely fast transfer rates and very large capacities.

3570 drive (a.k.a. Magstar MP)

The IBM 3570 cartridge, with its trapezoidal shape, is a completely different form factor than just about any cartridge out there. It was the first midrange tape drive that could mount midpoint; it also never leaves the cartridge. (In the tradition of borrowing old technologies, this one reminds me of an old cassette tape. You remember how pinchers were inserted into the tape, and the rollers pulled the tape along without removing it from the cartridge? This mechanism is reminiscent of that.) The tape has a relatively slow transfer rate, but transfer rate is not the most important factor in the market that this tape is aimed at. It is a potential nearline solution, because it has a total time-to-data of less than 30 seconds—not that much slower than the optical drives that were available at the time it was released.

8 mm (8x0x) drives (end-of-lifed)

This family of drives was originally made only by Exabyte, although other manufacturers eventually made them as well. (This category does not include the AIT or the Mammoth drives, which are covered separately.) The first drive in this family was the 8200, followed by the 8500 and 8505. These drives have a poor reliability history. Insiders will tell you that it is because the mechanisms that go into

these drives are the same mechanisms that go into Sony 8 mm camcorders. At one time, they actually were made on the same assembly line.

> **Tip ✎**
>
> These drives were so much like camcorders that back in the day, we actually used consumer-grade video tapes as backup tapes. Although we cursed our cheap purchasing department, they actually worked just fine.

9840 drives

This line of Sun/StorageTek drives uses a half-inch 9840 cartridge and includes the 9840A, B, and C. At first glance, they have smaller capacities and throughput rates than other drives listed in this chapter, and yet they cost more. There are two reasons for this. The first is that, like the IBM 3xx0 drives, these drives were designed for mainframe use and are intended for a 100 percent duty cycle. The second reason is that they tend to have very quick time-to-data rates. If you need a very quick-acting tape drive that has a 100 percent duty cycle, this drive is for you.

9940 drives

The 9940 drives are also from Sun/StorageTek and use a half-inch 9940 cartridge. They offer more capacity and throughput than the previous 9840 generation.

T10000 drives

The next generation of drives from Sun/StorageTek uses a half-inch 10000 cartridge that offers more capacity and throughput than the 9940 family.

AIT drive

The AIT drive is Sony's attempt to completely reengineer the 8 mm drive. These drives fit into a 3.5-inch half-height form factor. The tapes are 8 mm high, but that's where the similarity with the old 8xx0 drives ends. Sony invented a new type of media just for the drive, called Advanced Metal Evaporative (AME), which consists of an evaporated metal recording layer covered by a protective layer and lubricant. This new media type reportedly has superior recording and magnetic retention capabilities over the standard magnetic particle tape. The tape also contains an EEPROM called Memory In Cassette (MIC). This EEPROM contains historical information about the tape, and potentially could be used to partition the tape into multiple logical volumes, although no vendors have taken advantage of this functionality. The AIT drive cannot read traditional 8 mm tapes. The drive did deliver on its promise of capacity and throughput, at a relatively low price point, and was designed to compete with the original DLT market.

DDS drive

Originally put out by HP, the Digital Data Storage (DDS) drive borrowed the format from the DAT market. Just for the record, it is not proper to call a DDS drive a DAT drive, because DAT refers to *digital audio tape*. Many people (and some vendors) still call DDS drives "DAT drives," even though it's about the same as calling an 8 mm drive a camcorder. Very few people will notice or care if you make this common mistake. It probably would be easier to get people to stop saying "PIN number." DDS drives are one of the least expensive and slowest drives in the open-systems market. They work, they're inexpensive—but they are slow.

They’re also quite popular, as they were the only midrange drive under $1,000 for a long time.

DLT drives (end-of-lifed)

DLT stands for Digital Linear Tape, and these drives were originally developed by Digital Corp., based on its TK-50 and TK-70 lines. The company kept the same basic media format and redesigned the drive that used it. (The first DLT drives were actually able to read the old TK tapes.) Two years later, it improved the design with what would come to be known as the DLT 2000, with twice the capacity and 60 percent greater throughput than its nearest competitor. The only problem was that no one outside Digital’s installed base was buying the drives. In 1994, it decided to sell the technology to a hard-drive manufacturer, Quantum Technology. The rest, as they say, is history. DLT drives then dominated the midrange storage market for quite some time.

DLT-S drives (aka Super DLT)

Quantum improved on the original DLT product with what it called the Super DLT. The early Super DLT drives could actually read older DLT media, and this was a great competitive advantage against any other drive, because most companies had a huge pile of DLT tapes with old backups on them. Since the Super DLT actually used a completely different read-write head, Quantum achieved backward compatibility with a separate read head for the older media. Quantum now calls this line DLT-S.

DLT-V drives (aka Value DLT)

Some entrepreneurs felt there was still a demand for lower-end DLT drives, so Quantum licensed DLT technology to a

company called Benchmark, which began making a value-based line of DLTs. These drives offered smaller capacities and slower throughput rates than Quantum's Super DLT drives, but it did so at a smaller price point as well. The Benchmark line was successful, so Quantum bought Benchmark and now markets these drives as its value DLTs, or DLT-V.

DTF drive

This is Sony's entrance into the high-capacity, high-speed market. These drives offer high capacity and aggressive throughputs, and the media is somewhere between the size of an LTO tape and a VCR tape. You typically won't see DTF drives in anything but a Sony tape library. Most libraries support similar-sized media such as 4 mm, 8 mm, or half-inch media, as they can easily retool a library by simply swapping out the drives and tapes. This unfortunately leaves DTF media out in the cold due to its form factor.

LMS NCTP drive

Phillips LMS originally made this drive, but Phillips LMS was sold to Plasmon. The Laser Magnetic Storage (LMS) NCTP drive can read or write to 3480 and 3490E cartridges. Although they have not gained wide acceptance with other automation vendors, they have moderate throughput rates and capacities, and Plasmon does offer its own line of tape libraries using these drives. (They have a similar size issue to the DTF media.)

LTO drives

Linear Tape Open drives are the product of the LTO Consortium between HP, IBM, and Quantum, and are the most popular midrange tape drives on the market today.

Some people seem to like that there are multiple companies that make these drives, which is something that isn't true of the Quantum DLT drives. Competition has done its job, with each member of the consortium working hard to make its version of LTO the best, while making sure its LTO was able to read tapes made by other members of the consortium.

Tip ✎

You've got to hand it to Quantum. The LTO Consortium, originally started by HP, IBM, and Seagate, was intended to compete with Quantum's DLT line. Seagate spun off Certance and gave it the LTO line, and then Quantum acquired Certance. Now Quantum is one of the members of the consortium that was designed to take it out. In addition, in 2006, Quantum acquired the only major competitor to its tape library business that wasn't owned by a major OEM. IBM owns its tape library business, and Sun owns what used to be StorageTek. The only remaining major competitor was ADIC, and it is now owned by Quantum.

LTO tapes are half-inch cartridges, and LTO drives offer very fast transfer rates and large capacities at a relatively moderate (although not inexpensive) price point. Most LTO drives also offer some degree of variable speed, and are able to step down to roughly half their original speed in order to keep up with slow incoming data rates.

Mammoth drive (end-of-lifed)

The Mammoth drive was Exabyte's attempt to go on its own. It listened to the complaints from its customers about the original 8 mm line but was unwilling to completely

reject the format. It believed that the failure of the original line was due to the consumer-grade components produced on the camcorder assembly line. It decided, therefore, to go on its own and produce its own mechanism. It increased the form factor so that the parts wouldn't be so cramped, and it upgraded several key parts of the design. This was a complete redesign, including everything from metal thickness to a different kind of material for the capstan rollers. The Mammoth drives offered much greater capacities and throughput rates than the original 8 mm drives, and Exabyte sold quite a few for a while. However, it eventually pulled the drive from the market due to lack of demand.

MLR 1-3 drives

Just as the DLT drive was a modified TK 70, the MLR drives are a completely new drive based on QIC media. MLR drives offer moderate capacities and speeds but have not gained wide-spread acceptance in the data center or from automation vendors.

VXA

Originally developed by Ecrix, the VXA technology was acquired by Exabyte. VXA cartridges offer an option to those looking for an inexpensive tape drive with reasonable capacity and performance. VXA makes many very strong claims about the reliability of data written to VXA tapes, including testimonies from people who were still able to read their tapes after very bad things happen to those tapes. Although Exabyte offers some tape libraries using VXA, VXA has not been adopted by major automation vendors. Again, it is probably the difference in the form factor.

Optical Drives

Optical drives fit a niche in between the inexpensive nature of tape and the throughput and flexibility of disk. They excel in the areas of reliability, flexibility, duty cycle, removability, and time-to-data. Their challenges are in the areas of throughput, capacity, and cost.

Optical drives offer the removability of tape, but their capacities and throughput rates have generally lagged behind those of tape. Their greatest advantages over tape have been their very quick time-to-data rates and their long-term reliability. Although inexpensive disk is now easily the winner in the time-to-data category, most disk systems are not designed to be portable. Therefore, optical systems are an easy choice for environments that require very quick time-to-data rates along with removability. In addition, if you need to reliably store data for long periods of time, optical is the clear winner. (Some disk units are even competing for that category as well, but many feel it's a stretch. We'll see what happens.) If you need a removable media with random accessibility that can reliably store data for long periods of time, optical tape drives are hard to beat.

> **Warning ☠**
>
> Not all optical media is created the same when it comes to long-term retention. Do your research.

This section is divided into two main subsections: optical recording methods and optical formats. Throughout, a recording *method* refers to how the data is physically represented on disk, and a recording *format* refers to what type of drive may use these recording formats. For

example, the most common recording method is the phase change method. There are several recording formats that use this, such as CD-RW, DVD-RW, DVD-RAM, and DVD+RW.

Optical Recording Methods

Most people understand that traditional disk drives record digital (binary) data by polarizing sections of the disk. With traditional optical recording methods, *pits* (or holes) in the *land* (or surface) of the disk represent the binary data. Historically, the land was the flat surface of the disk, and the pits were actual holes burned into the land. When the laser reads the disk, pits don't reflect as much light as the land, and this is translated into binary data. Many newer recording technologies do not produce *actual* pits. The land is composed of material that is sensitive to high-powered lasers. When a laser is applied to a certain area, it changes the reflective properties of that area so that it *appears* to be a pit in the land. (Although they aren't actually pits, they still are referred to as such.) The magneto-optical recording method is a hybrid of magnetic and optical technologies, thus its name. Binary data is represented by realigned magnetic particles, as with a traditional disk or tape drive. A laser is used during the recording process, and a laser is used when reading the drive as well. (More on this later.)

Magneto-optical recording method

With magneto-optical (or MO), the magnetic recording layer is heated by a laser, which makes it easier to polarize. The data then is recorded using traditional magnetic recording techniques, although the heated nature of the recording surface allows precise control over the

magnetized areas. Once the recording layer cools, it is less susceptible to magnetic degradation. An MO disk is erased by passing the high-powered laser over it again and writing all zeros to the disk. This two-step operation usually requires two passes to complete, but some MO drives have figured out how to do it in one pass. MO drives range in size from a few hundred megabytes to 9.1 gigabytes. MO media traditionally has been permanently contained within a cartridge that looks similar to a very big 3.5-inch floppy. (You do remember floppies, right?)

At a time when CDs offered only a few hundred megabytes, and affordable DVD writers didn't exist, MO drives offered the first affordable "optical" recording method for larger amounts of data. Although it was actually a magnetic recording, it stored data longer than traditional magnetic recordings because the media had to be heated to change its magnetic properties. The result was that MO ruled the archive world for quite a few years. However, the introduction of the phase change recording method changed all that.

Phase change recording method

Phase change recording is the only truly optical recording method, as compared to MO, which is a mix of both optical and magnetic recording. Disks recorded with this method contain a special recording layer that changes between a crystalline and amorphous state when heated by a laser, which is why it is called "phase change." The amorphous state is less reflective than the crystalline state, and sections of the disk that are in the amorphous state appear as pits when the disk is read. A higher-powered laser returns the entire disk to the crystalline state, thus erasing the data. As of this writing, it is used in all recordable and rewritable CD, DVD, and UDO drives.

The phase change recording method has some advantages over the MO recording method, starting with the fact that a disk can be erased in a single pass. Secondly, the nature of laser recording allows for lasers with smaller and smaller diameters, whereas the lasers themselves become more powerful. We started with 780 nm pits with CDs and followed with DVDs with 630 nm pits. Now the blue lasers used by UDO drives are only 405 nm wide. The small width of the lasers allow for tighter tracks with bits that are closer together, allowing for more data on the disk. Couple that with higher-powered lasers that can read and write multiple layers, and you can understand how we've gone from 650 MB to 30 GB on the same diameter disk. The result is that, at this point, devices using phase change recording have really taken over the market.

Dye polymer recording method

This recording method has been adopted by a few manufacturers and uses a special die that bubbles when heated. The bubbled areas have different reflective properties than the nonbubbled areas, and appear as pits when read by the laser. Again, a higher-powered laser resets the dye to its original state, thus erasing the data.

WORM recording methods

Slight variations to the recording methods just discussed result in write-once-read-many recordings that create an archive that cannot be deleted, changed, or overwritten. A number of companies use WORM recording methods to create archives that can be used as proof of things that happened in the past, because you can prove that they haven't been changed.

CD-R, DVD-R, DVD+R, and UDO all use a variation of the phase change recording method to create WORM disks. There are also WORM variations of the MO recording method. There's even WORM tape. For a recording to be WORM, the media must be designed so that it can be changed only once. For example, the popular ablative recording method uses a tellurium alloy as a recording layer. This alloy has a low enough melting point that the high-power laser actually does make physical pits in it. Once these pits are in the disk, they cannot be removed.

Optical Recording Formats

The following is an overview of recordable and rewritable optical formats. They will be covered as generally as possible, because the details change every time you look at them.

CD recording formats

Recordable and rewritable CDs are quite popular. They have a small capacity (about 700 MB) and a relatively slow transfer rate (1–2 MBps), but everyone can read a CD these days. Therefore, they're useful for data interchange if nothing else. Here are the two types of CD recorders and how they work:

CD-R

> CD-R recorders use one of the WORM recording methods to create read-only CDs that can be read in any normal CD-ROM or audio CD player, making this a very popular format. CD-R recorders usually are used to create permanent archives or bootable CDs.

CD-RW

CD-RW recorders use the phase change recording method to produce CDs that can be overwritten if desired. CD-RW CDs are readable only in “multiread” CD-ROM drives. Most CD-RW drives also can produce CD-R CDs that can be read in older CD-ROM drives. The biggest disadvantage to this format is its speed when compared to CD-R.

DVD recording formats

Depending on who you ask, *DVD* stands for *digital versatile disk* or *digital video disk*. DVD has gained a lot of popularity in recent years, and it brings a lot to the table. There are also a lot of automation vendors that offer rewritable DVD libraries. The different recordable formats can fit between 2.6 GB and 4.7 GB per side, for a total of 9.4 GB for a dual-layer, dual-sided disk. The transfer rate (5–20 MBps) is still very slow compared to modern tape drives, but most recordable DVD formats support random access, so they have a very quick time-to-data.

At one time, the user was required to carefully select a format for the long term. There are two WORM formats (DVD-R and DVD+R) and three rewritable formats (DVD-RW, DVD+RW, and DVD-RAM). At this point, however, most recordable DVD drives can read or write any of the five competing formats.

The DVD formats, described below, are sponsored by two different groups: The DVD Forum (http://www.dvdforum.com) and the DVD+RW alliance (http://www.dvdrw.com). The DVD Forum was the original alliance of DVD manufacturers, but their DVD drives and media are no more or less official than those provided by the DVD+RW alliance. Drives with a hyphen or dash in their name (DVD-RAM, DVD-R, and DVD-RW) come from

the DVD Forum, and those with a + in their name (DVD+R, DVD+RW) come from the DVD+RW Alliance.

DVD-RAM

DVD random-access memory (DVD-RAM) was the first readily accessible, affordable, rewritable DVD format, using a phase change recording method such as CD-RW. DVD-RAM is probably the least popular of these formats, perhaps due to its lack of compatibility with regular DVD drives.

DVD-R

DVD-recordable (DVD-R) is the only DVD format to use the dye polymer recording method. DVD-R drives use WORM media that can be written to only once. DVD-R's big plus is that the disks that DVD-R drives produce can be read easily in any other DVD drive. That is because they appear identical to a "normal" DVD disk.

DVD-RW

DVD-rewritable (DVD-RW) is the second rewritable format to be endorsed by the DVD forum, and it uses the phase change recording method. It is also readable in most DVD-ROM and drives, but not necessarily readable in regular DVD players. It is more expensive than DVD-RAM, but it does not have DVD-RAM's drawbacks.

DVD+R

DVD-recordable (DVD+R) is the newest recordable DVD format, and it was proposed by Sony, HP, Ricoh, Yamaha, and Phillips. It uses the phase change recording method and is able to create either a sequential or random access disk.

DVD+RW

DVD-rewritable (DVD+RW) is the newest rewritable DVD format, and it was also proposed by Sony, HP, Ricoh, Yamaha, and Phillips. It likewise uses the phase change recording method and is able to create either a sequential or random access disk.

Unix Backup & Recovery had a compatibility chart for the various DVD formats. However, I found through interviews and my own experience that your mileage will vary greatly. Generally speaking, the DVD-R format is the most compatible with other players. I know that's what I burn when I want to make sure the other person can read the DVD. Other people have told me that they haven't had near the problems that I have had with incompatibility. That's why I say your mileage will vary. Do your own testing to be sure.

Magneto-optical recording format

MO drives use the MO recording method and are readily available from a number of vendors. There is also a big line of automated libraries that support MO drives and media. This level of automation, combined with its low cost, used to make MO the choice for nearline environments. The challenges of MO, including a limited capacity (9.1 GB) and a multipass rewrite, caused people to start to look elsewhere. As of this writing, it appears that DVD and UDO have gained significant ground in this area.

UDO recording format

Ultra Density Optical, or UDO, is a 5.25-inch rewritable optical disk designed by Plasmon that also supports both rewritable and WORM media. As of this writing, it fits 30 GB on a single cartridge using blue lasers and phase

change recording. Cartridges that store 60 GB and 120 GB are on the way.

While UDO is similar to CD and DVD drives designed for the consumer, it was designed with the data center in mind, with more expensive, heartier components than you can use in a consumer-grade appliance. UDO also uses an eight-layer disk, which is twice the number of layers present in a typical consumer-grade DVD.

Automated Backup Hardware

So far, this chapter has covered only the tape and optical drives themselves. However, today's environments are demanding more and more automation as databases, filesystems, and servers become larger and more complex. Spending a few thousand dollars on some type of automated volume management system can reduce the need for manual intervention, drastically increasing the integrity of a backup system. It reduces administrator frustration by handling the most common (and most boring) task associated with backups—swapping a volume.

There are essentially three types of automated backup hardware. Some people may use these three terms interchangeably. For the purposes of this chapter, these terms are used as they are defined here:

Stacker

> This is how many people enter the automation market. Stackers get their name from the way they were originally designed. Tapes appeared to be "stacked" on top of one another in early models, although many of today's stackers have the tapes sitting side by side. A stacker is traditionally a sequential access device, meaning that when you eject tape 1, it automatically puts in tape 2. If it contains 10 tapes, and you eject tape 10, it puts in tape 1. You cannot tell a true stacker to "put in tape 5." (This capability is referred to as *random access*.) It is up to you to know which tape is currently in the drive and to calculate the number of ejects required to get to tape 5. Stackers typically have between 4 and 12 slots and 1 or 2 drives.

Many products that are advertised as stackers support random access, so the line is slightly blurred. However, in order to be classified as a stacker, a product must support sequential-access operation. This allows an administrator to easily use shell scripts to control the stacker. If you purchase a commercial backup product or use an open-source product that can control a tape library, you have the option of putting the stacker into random-access mode and allowing the backup product to control it. (Control of automated backup hardware is almost always an extra-cost option in commercial software.)

Library

This category of automated backup hardware is called many things, but the most common terms are “ library,” “autoloader,” and “jukebox.” Each of these terms connotes an addressable group of volumes that can be automatically loaded via unique volume addresses. This means that each slot and drive within a library is given a location address. For example, the first slot may be location 0000, and the first drive may be location 1000. When the backup software controlling the library tells it to put the tape from slot 1 into drive 1, it actually is saying “move the volume in location 0000 to location 1000.”

The primary difference between a library and a stacker is that a library can operate only in random-access mode. Today’s libraries are starting to borrow advanced features that used to be found only in silos, such as import/export ports, bar code readers, visual displays, and Ethernet ports for SNMP monitoring. Libraries may range from 12 slots to 1000 or more slots. The largest libraries even offer pass-through ports, which allows one library to pass tapes to another library. (This is usually a

standard feature in silos.) Some libraries can expand with extra cabinets that actually become part of the base chassis; the robot track is extended into the next cabinet.

Silo

Since many libraries now offer features that used to be found only in silos, the distinction between the two has blurred. The main distinction between a silo and a library used to be whether or not it allowed multiple hosts with disparate backup applications to connect to the same silo. However, libraries now offer this feature through something called *partitioning*. Therefore, many people just use the term *silo* to refer to a really large tape library—and it's hard to correct them.

Another way to subdivide the different types of automated backup hardware is how they allow you to expand them, if they allow that at all. Automated backup hardware can be divided this way into three different categories:

Nonexpandable

These types of libraries cannot be expanded beyond their base configuration. While this is most common in smaller units, there are plenty of large, nonexpandable libraries as well.

Connectable

A connectable tape library can be expanded by buying a second library and connecting the two. Each library has its own distinct robot, but they are allowed to share media with each other using pass-through ports. The big advantage to this method is that you can disconnect the different libraries and use them as separate libraries at a later date.

Expandable

An expandable library allows you to add additional capacity without adding additional robotics. The original robotics simply expand to handle the additional capacity. The big advantage to these types of libraries is that you can buy exactly the amount of capacity when you need it —and only when you need it.

Disk Targets

As mentioned in section "Tape Drives Must Be Streamed" earlier in theis chapter, properly streaming a drive increases both its throughput and reliability. The tactic that most backup software products used to stream drives was to send multiple backup jobs simultaneously to the same tape drive, a technique called multiplexing or interleaving. This technique helps backups but can have a negative impact on the restore of a single backup; the backup software reads the entire tape and disregards the data that it doesn't need. A few backup software products solve the streaming issue with *disk staging*, in which backups are first sent to disk before they are sent to tape.

If you want to speed up backups and restores, you should buy enough disk to hold all on-site backups.

With the advent of lower-priced ATA-based and SATA-based disk arrays, many backup software vendors are now adding disk staging features to their products, allowing many more people to take advantage of disk-based backups without switching from a traditional backup architecture. Augmenting your tape library with some type of disk has now become commonplace; the only problem is choosing among the various ways in which to do so. The following paragraphs should help.

You can augment your traditional backup system, illustrated in Figure 9-6, with disk. The first two options are called *disk-as-disk* because they are disk drives behaving as disk drives; they aren't emulating tape. In a *SAN disk-as-disk* configuration (see Figure 9-7), a disk array is connected to one or more backup servers via a SAN, and a disk volume is assigned to each server. Each backup server usually then puts a filesystem on that volume, and backups are sent to

that filesystem. (Some backup software packages can back up directly to a raw volume, but most cannot do so.) In a *NAS disk-as-disk* architecture (see Figure 9-8), the disk resides behind a filer head that shares filesystems via NFS or CIFS; backups are sent to those filesystems. The next two options use virtual tape libraries, in which disk systems are placed behind an appliance running software that allows the disk array to emulate one or more tape libraries. Figure 9-9 illustrates *standalone VTLs* that sit next to a physical tape library and emulate another tape library. Once you back up to a standalone VTL, you must use the backup server to copy its backups to physical tape if you want to be able to send them off-site. An *integrated VTL* (see Figure 9-10) sits between a physical tape library and a backup server, and it emulates the physical library it is sitting in front of. The backup server backs up to the integrated VTL, and the VTL then copies the virtual tapes to physical tapes without using the backup server. Finally, virtual tape cartridges are an interesting hybrid between virtual and physical tape.

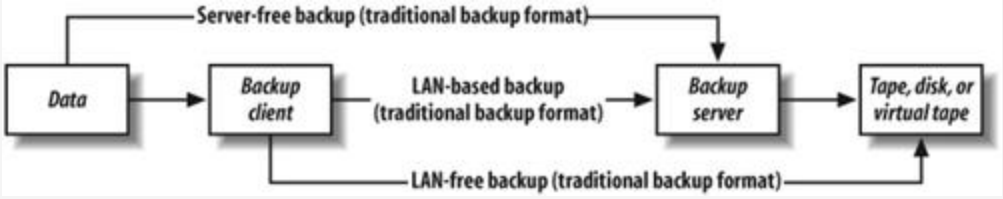


Figure 9-6. The traditional backup architecture

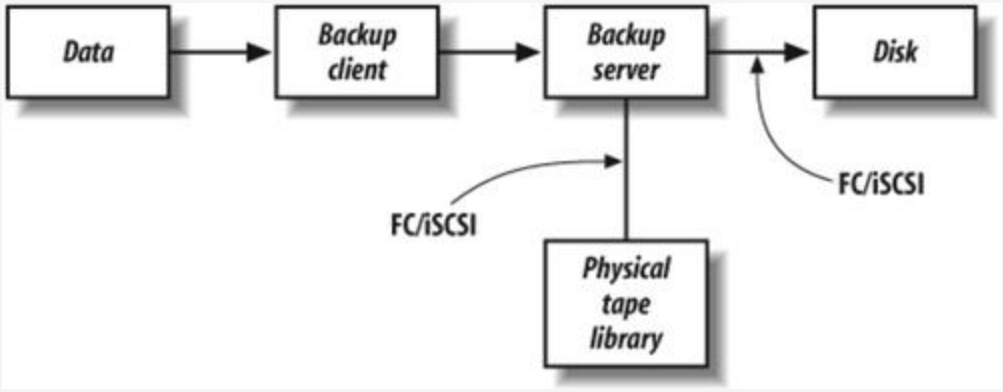


Figure 9-7. SAN disk-as-disk

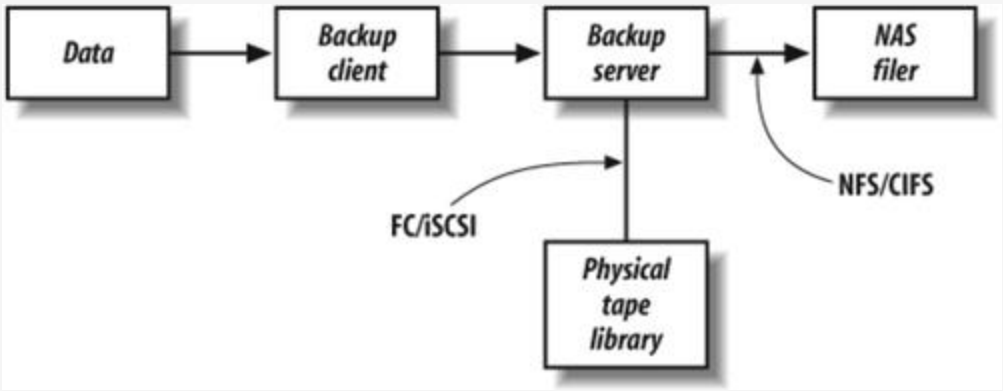


Figure 9-8. NAS disk-as-disk

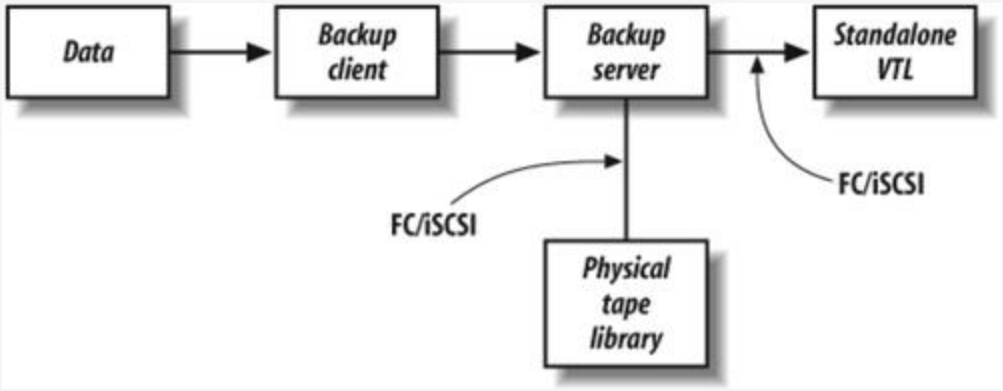


Figure 9-9. Standalone VTL

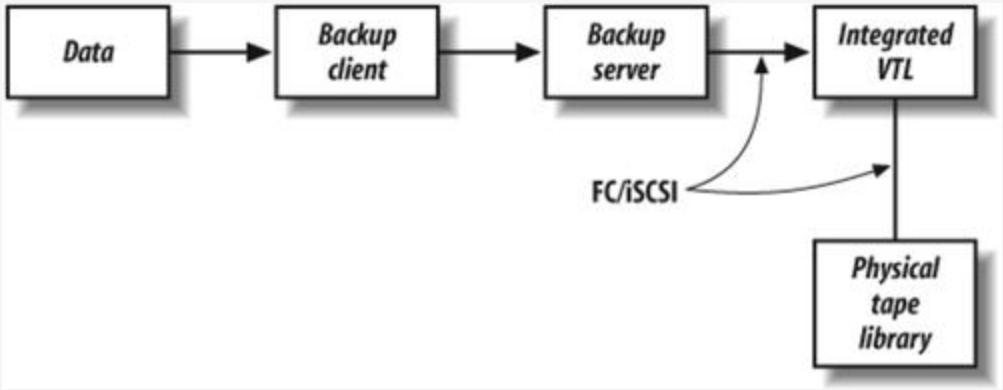


Figure 9-10. Integrated VTL

Disk-As-Disk Targets

When backup software backs up to a disk-as-disk system, it knows it is disk, and it typically creates a file within the filesystem. As mentioned previously, some backup software

packages are able to back up directly to a raw disk device, but this is rare.

Advantages of disk-as-disk targets

The biggest advantage that disk-as-disk targets have over most VTL targets is their cost. Most disk-as-disk systems cost less per gigabyte than VTL systems because you're not paying for the value of the VTL software.

Some people save even more money by redeploying an older, decommissioned array as a disk-as-disk target. Of course, decommissioned arrays are often end-of-life units without service contracts, and those service contracts should be resumed if you're going to use the unit in a production system. Since service contracts on older equipment can be quite expensive, be sure to compare the cost of resuming the contract to the purchase of a new system with a contract already included.

Another advantage of disk-as-disk backup targets is that most backup software companies aren't currently charging you to back up to them. We will discuss in the disadvantages section how this is changing.

The final advantage of disk-as-disk targets is their flexibility, and this flexibility may come into play if you plan on moving away from a traditional backup architecture. For example, if you're looking at adopting a de-duplication backup system, a CDP system, or near-CDP system, all these options require a disk-as-disk target. These advanced commercial data protection options are beyond the scope of this book, so the following are some very quick definitions. A de-duplication backup system attempts to eliminate redundant blocks of data at the client level and transmits only new unique blocks of data to the backup server. A near-CDP system uses replication and very frequent

snapshots to create many points in time to recover from, and a true CDP system backs up every block of data as it changes but stores each block in a continuous log, allowing you to restore to any point in time. There's more information about these types of backup products in Chapter 8.

Disadvantages of disk-as-disk targets

At this writing, most backup software products do not require additional licenses to back up to a disk-as-disk target, but this is changing rapidly. Backup software companies are starting to charge for what used to be free, and this trend is expected to continue. The backup software companies defend this move because they are adding additional functions to their backup software.

Another disadvantage of disk-as-disk backup devices is the nature of filesystems. Files are written, opened, changed, and stored back to the same place. Often, the new file doesn't fit in the same place where the old file was, so a portion of it gets written to the original location while another portion is written somewhere else on the disk, resulting in fragmentation. The more files you add, delete, and add back, the more fragmented the filesystem becomes. The way a backup system uses the disk will result in significant fragmentation over time that will degrade performance. (Later, we will explain why VTLs don't have this limitation.)

Yet another disadvantage of disk-as-disk backup targets is that some backup software products don't back up to filesystems as well as they back up to tapes. For example, backup software products know exactly what to do when a tape fills up, but they're not always sure what to do when a filesystem fills up. Some backup products require you to point disk-as-disk backups to a single filesystem. When that

filesystem fills up, all the backups fail—even if there is another filesystem with adequate capacity.

Another challenge with disk-as-disk backup targets is how to create off-site backups. Although it's a best practice to copy the original backup to tape and then ship that tape off-site, most people don't do this with their tapes—they simply eject the original tape and send it off-site. You can't do this with a disk array; therefore, you need to learn how to copy disk-based backup data to tape and how to automate the process. Automating this can range from extremely easy to extremely difficult, depending on the backup product you use, and it may require the purchase of additional software from your backup vendor. Whatever method you choose in order to get the data from disk to tape, remember that the data is now moving twice, where before it only moved once (if you were ejecting originals). You'll need to budget time for the data to make that second move.

> **Tip ✎**
>
> Be sure to read the section "How do you eject virtual tapes?" later in this chapter to see whether VTLs have the same problem.

One final disadvantage of disk-as-disk targets is the lack of hardware compression. Hardware compression can increase the speed and capacity of a backup device by as much as 100 percent. Some disk-as-disk products do support *de-duplication*, which should not be confused with compression. See "Disk Features to Consider" later in this chapter for more information on de-duplication.

SAN disk-as-disk targets

A SAN disk-as-disk target (see Figure 9-7) is simply a disk array connected to the SAN and attached to one or more backup servers. The backup server puts a filesystem on the array and writes to that filesystem. The advantage over a NAS disk-as-disk system, of course, is the superior write performance typical of a high-end SAN disk array compared to an IP-based NAS filer.

However, when you use a disk array as your backup target, you replicate into your secondary storage all of the provisioning issues of your primary storage. All of the hassle with associating disks to RAID groups, RAID groups to servers, and volumes to filesystems now needs to be done on the back end of your backup system. This problem is compounded when you have multiple backup servers. When using a tape library or VTL, most backup software packages know how to share these devices. However, if you're using a SAN disk-as-disk target with multiple backup servers, you're usually going to need to decide how big each backup server's volume needs to be and to allocate the appropriate amount of space to each backup server. (Some backup software packages are capable of dynamically sharing disk, and this removes a lot of the provisioning issues.)

NAS disk-as-disk targets

A NAS disk-as-disk target (see Figure 9-8) removes many of the provisioning issues of a SAN disk-as-disk target by putting the disks behind a NAS head, making a giant volume and sharing that volume via NFS or CIFS. Generally speaking, such systems are also easier to maintain than traditional disk arrays. However, remember that easier management comes with a cost. Both the filer head and filer OS add to the cost of the system, and performance is limited to the throughput of the filer head. Depending on

the size of your backups, though, performance may not be an issue. Also, if you're a NAS shop with many other filers, a NAS disk-as-disk target makes perfect sense—especially if you are going to examine a near-CDP system.

Clustered Filesystems As Targets

Clustered filesystems could address some of the disadvantages of disk-as-disk targets. They help with provisioning by not requiring you to use a volume for each backup server, without suffering the single-head limitation of a typical NAS system. The challenge as of this writing is that clustered filesystems are still considered bleeding edge by many. While they're solving many problems for many people, they also come with limitations that may cause other problems for backup systems. For example, some of them offer very good performance for hundreds of simultaneous streams of data, but poor performance for any individual stream. Clustered filesystems are also typically more expensive than the other options we're considering here. If these limitations are removed, clustered filesystems may someday be an option as well.

Disk-As-Tape: Virtual Tape Libraries

Virtual tape libraries are the alternative to disk-as-disk targets. They are also called *disk-as-tape* units because they are disk arrays emulating tape. Many people believe that VTL software is something else to pay for in addition to the disk array that they're already buying, and they're unsure of why they should do that. This section should help answer that question. It begins with a discussion of the advantages and disadvantages of all VTLs, and then explains the difference between standalone and integrated VTLs and the

advantages and disadvantages of each. Finally, it describes the features that you should consider when deciding which VTL to purchase.

Advantages of VTLs

The main advantages of a VTL over a disk-as-disk target are ease of management and better performance. As mentioned in the previous section, a disk-as-disk target requires all of the usual provisioning steps of standard shared storage arrays. In contrast, you tell a VTL how many virtual tape drives and virtual cartridges you want it to emulate, and you're done with provisioning. The VTL software automatically handles all the provisioning, allocating the appropriate amount of disk to each virtual cartridge and dynamically giving access to that cartridge to each backup server that needs to use it.

> **Tip ✎**
>
> Not all VTLs eliminate provisioning to the same degree. This is an important thing to consider when evaluating a VTL.

Another important management advantage of VTLs is how easy it is to share VTLs between multiple servers and applications. If you need to share a VTL between multiple backup servers running the same software, you can use the built-in library sharing capability that most commercial backup products already have. If you need to share a VTL between multiple servers that don't support dynamic sharing (such as with an open-source product or when using it with multiple products that can't share with each other), you can just partition the VTL into multiple smaller VTLs, assign a certain number of virtual cartridges to each

VTL, and associate each VTL with a different backup server. You can also do this if you don’t want to use the library sharing software offered by your backup vendor—just give each backup server its own tape drives. Both of these scenarios are much easier than what is required to share a disk-as-disk target between multiple backup servers.

To understand the performance advantages of VTLs, you must first think about how backup applications write data to tape. Once a backup application starts writing to a tape, it typically continues writing to that tape until it hits physical end of tape. It will append to a tape even if some of the previously written data has already expired. Once the backup application hits PEOT, the tape is considered full, and most backup applications leave everything on the tape until all backups on that tape have expired. Then they expire the whole tape and write to it from the beginning of the tape. Other backup applications wait until a certain percentage of the backups on a tape have expired and then reclaim that tape by migrating all of the nonexpired backups to a second tape and then expiring (and subsequently overwriting) the first tape. The point is that you do not overwrite portions of a tape; tapes simply don’t work like that.

This is very different from how backup applications write to a filesystem. They tell the operating system that they want to write to a certain filename and then start writing data to that file. Each backup gets its own file. When that file expires, it is deleted. The backup application has no knowledge of how this data is actually written to disk. Underneath the covers, of course, the bytes of any given file are fragmented all over the disk. This fragmentation results in a loss of performance.

VTLs treat disk like tape. They know they are being used for backups and archives, and they know how backups and archives behave. This allows them to eliminate fragmentation by writing backups to contiguous sections of disk. The blocks allocated to a given tape stay allocated to that tape until the backup application starts overwriting that tape, at which point the VTL can again start writing to contiguous sections of disk—just like we write to tape. Some VTL vendors even control the RAID volumes, allowing them to make sure that a given RAID group is only being written to by a single virtual tape. Think of how well a disk can perform if it is only writing/reading for one application at a time that is always writing/reading with contiguous sections of disk.

Another key performance difference is the use of clustering in some VTLs. While clustered filesystems are still the exception and not the rule, this is not the case with VTLs. Several VTLs offer the idea of multiple data movers, where each data mover adds additional capacity and throughput. The use of clustering along with their customized way of writing to disk should explain why the fastest filesystems write in hundreds of megabytes per second, but the fastest VTLs write in thousands of megabytes per second.

VTLs have other advantages, as well. With one exception (covered in the disadvantage section), your backup software and your backup operators and administrators can use all of their existing technology, processes, and procedures when using a VTL for backups. This means that everything works exactly as it would if you were using a physical tape library (PTL). That is not the case with disk-as-disk targets, where backup software can behave quite differently.

Some VTLs also support compression, allowing you to fit more backups on the same amount of disk. (This should not

be confused with data de-duplication, which is covered later in the chapter.) Whether you purchase and use the compression feature of your VTL or not will be based on which VTL you have and how you are using it. Unfortunately, many VTLs use in-band software compression that saves space, but results in a significant performance hit—as much as 50 percent. If your backup speed is throttled by the speed of your clients and/or your network, you may not see this performance hit. You will probably see it in local or LAN-free backups, though, as their speed tends to be throttled by the backup device. Some VTL vendors support hardware compression that doesn't impact performance. (They accomplished this by using the same type of chip that is used in front of tape drives.)

Disadvantages of VTLs

The first disadvantage of VTLs that many people mention is cost. People believe that if a disk array costs X, a disk array made to look like a VTL is going to cost X + Y. Oddly enough, disk-as-disk units have roughly the same price range as VTLs, which means it's basically not fair to say that VTLs always cost more than disk-as-disk units. Your cost is based on which VTL you purchase and which disk-as-disk target you compare it to. Most VTLs use capacity-based pricing, meaning it costs $X/GB. As of this writing, the most expensive VTL is three times the price of the least expensive VTL, so it pays to shop around. Finally, if your VTL supports data de-duplication, it can make the VTL even cheaper.

Another cost issue that people have with VTLs is backup software licensing. If you purchase a VTL to sit next to your existing tape library, you are probably going to need to buy an additional tape library license...for a library that's not

really there. This, of course, adds to the price of the VTL. How much you actually pay is based on how your backup software charges for PTLs and/or VTLs. Some backup software products have a single license for all tape libraries, while others charge for the number of slots or the number of drives. Some also have capacity-based pricing for VTLs. Therefore, consider how your backup software charges for PTLs and VTLs when deciding how to configure your VTL. (When comparing VTLs to disk-as-disk targets, remember that backup software products are also starting to charge you to back up to disk-as-disk targets.)

How do you eject virtual tapes?

The answer to this question depends on whether or not you purchase a standalone VTL (see Figure 9-9) or an integrated VTL (see Figure 9-10). As discussed previously, one major advantage of VTLs is that they do not require any changes to your existing backup process or configuration. The one exception to this is if you are not used to copying your backup tapes and sending the copies off-site. Although it is not a best practice to do so, many environments eject their original tapes and send them off-site. This works fine with a PTL, but not with a VTL. Therefore, companies that eject their original tapes and wish to use a VTL must usually do one of two things: learn how to copy tapes or use an integrated VTL. Which of these two approaches is best for your environment will be based on your preferences.

Some say that the tape-to-tape copy method with standalone VTLs is the only proper way to create physical tapes from virtual tapes. It allows the backup software to control the copy process, therefore integrating the copy process into your normal reporting procedures. However, this method has two challenges. The first challenge is how

difficult it may be to automate this process, depending on which backup software product you are using. With some backup products, you may simply need to read a new section in the manual and do what it says. With others, you may need to purchase an additional license. Finally, some backup products actually require you to script this process. You need to determine the level of difficulty for your environment. The second challenge with this process is that many environments do not have enough time in the day and resources in their system to copy their backup tapes quickly enough. For many companies, it is all they can do to get their backups done in time for the vault vendor to pick them up. Of course, if you already know how to copy your backup tapes, and you have enough resources to do so, this disadvantage is a nonissue.

If the challenges of copying your virtual tapes to physical tapes concern you, you might consider an *integrated VTL*. There is a lot of FUD (Fear, Uncertainty, and Doubt) out there about integrated VTLs, so please read carefully about how they work. An integrated VTL sits between your backup server and your physical tape library. It inventories the PTL and represents its contents as virtual tapes in the VTL. For example, if you have physical tape X01007 in your PTL, virtual tape X01007 will appear in your VTL. Your backup software will then back up to virtual tape X01007. At some user-configurable point, virtual tape X01007 is copied to physical tape X01007. Then when your backup software tells the VTL to eject virtual tape X01007, physical tape X01007 appears in the PTL's mail slot. An important point is that physical tape X01007 looks just like it would if your backup software backed up directly to it. Your backup software thinks that it backed up to and ejected physical tape X01007, and in the end, that's what it did. Using bar code matching like this maintains the consistency between the backup software's media manager and the physical

tapes. Remember, however, that this method does not result in two copies of the tape. The virtual copy of the tape is deleted when you successfully create the physical copy.

> **Warning ☠**
>
> Some integrated VTLs allow you to have virtual tapes with bar codes that don't match physical tapes. Don't do this! Your backup software will ask for one tape, and you'll have to ask your VTL what tape that really is.

There are some challenges with this method as well. The first challenge is what happens when the copy from virtual tape to physical tape fails. If the reason the copy failed is that the actual tape is bad, you need to remove the tape, swap its bar code to a new tape, put the new tape in the PTL, and tell the VTL to try the copy again. (Of course, this works only if your bar codes are removable.) If this happens occasionally, you might not consider this a major disadvantage. However, if it happens every day, it could get quite annoying. It's also important to realize that the entire process is happening without the knowledge of the backup software, which means that if something does happen with a tape copy, the VTL will need to notify you of the problem. This, of course, means you have another reporting interface, which some would consider a disadvantage as well. Another potential problem is what would happen if the VTL put more data on the virtual tape than would fit on the physical tape. You would not be able to create a physical copy of this tape. Integrated VTL vendors make sure this doesn't happen by stopping way before the normal physical end of tape. Standalone vendors mention that this increases the number of tapes to purchase and handle, increasing your costs. Remember that if you

purchased an integrated VTL and decide not to use its tape copy features, it can act as a standalone VTL.

> **Tip ✎**
>
> As of this writing, some VTL vendors are working with some ISVs to allow the ISVs to control the copy process but still allow you to take advantage of the features of an integrated VTL. That would give us the best of both worlds.

Disk Features to Consider

The following section covers some major features to look at when considering the purchase of a disk target. Because most of the advanced features are being developed by VTL vendors, most of these features apply only to them.

Packaging

Since what VTL and NAS vendors sell is basically software running in some type of host or appliance, let’s consider how these types of systems are packaged. Some VTL/NAS vendors are software-only, which means that you can buy just their software to run on your own CPU and your own disk. Some VTL/NAS vendors sell you a VTL/NAS head. You use their software and their head, but you supply your own disk. Finally, some VTL/NAS vendors prefer to sell you the entire solution software—head, disk, and all. Software only and filer head vendors obviously allow you to redeploy an existing array, reducing your cost. Turnkey vendors cost more but have the fewest integration issues.

De-duplication

Probably one of the most important features to consider is the availability of de-duplication. De-duplication is a rather intensive process that examines each block of data as it comes into the device and attempts to determine if it's seen the block before. If it hasn't, it stores it. If it has seen the block before, it throws the block away and stores only a reference to it. Examples of obvious duplicate data that it would identify and store only once are listed here:

- The same file backed up from five different servers
- A weekly full backup when only 5 percent has changed (95 percent would be duplicate blocks from last week)
- A daily full backup of a database that doesn't support incremental backups (most of it would be duplicate blocks from the day before)
- Incremental backups of files that change every day, such as a spreadsheet that gets updated every day; only the changes would get stored every day

Consider a typical data center with a mix of database data and filesystem data, performing weekly full backups and daily incrementals. The average de-duplication system can reduce the amount of storage needed to store its backups by 20 to 1 or more. Those performing monthly full backups see a lower de-duplication ratio. You may see much higher numbers in vendor literature; do not make the mistake of comparing two vendors based on their de-duplication ratio claims. While some vendors may be better than others in eliminating redundant data, what matters is how well that vendor's algorithm works with your data. Be sure to benchmark the performance and de-duplication ratio of each system using your own data.

Replication

If you're storing only new unique blocks every time you run a backup, another interesting feature to consider is replication, or *cascading*, as it is sometimes called in the VTL world. This feature allows you to replicate one disk device's backups to another device. Remember that without data de-duplication (discussed later), what you will be replicating is the entire backup, and remember that most backups are not block-level backups. Even incremental backups take up roughly one to five percent of the amount of data being backed up, which means you will need to replicate one to five percent of your data center every night —a significant undertaking for many environments. Therefore, it may be possible to use this feature only within the campus—unless your device supports de-duplication.

Also keep in mind that the backups in the second device will not be considered duplicates by your backup software because they will have the same bar codes or filenames as the original tapes. How you go about using these tapes in a disaster or outage will depend on your backup software. You may need to replicate your backup catalog and start up another backup server in the new location. If your old backup server is still alive, you may be able to tell it that the primary tape library was simply moved to the new location. Make sure you figure out what to do before it happens.

Content-awareness

A *content-aware* disk target understands the backup formats being sent to it and extrapolates the original data from that format wherever possible. A simplified version of this would be taking a `tar` stream and turning it back into the files that are in the `tar` stream. If that's all a content-aware system did, it wouldn't be offering any additional features. However, if a disk target can turn the backups into the

backed up files, such a system may be able to do something else that we'll call *re-presentation*.

> **Tip ✎**
>
> Content-aware products are typically aware only of the top three to five commercial backup products and may not understand open-source products or products with a small market share.

Re-presentation

If your disk target is content-aware and has extrapolated the original data from its backup format, it can potentially do a number of interesting things by re-presenting the data back to you. Some products present the data using a web page with directory trees whereas others present the data via an NFS or CIFS mount. In all cases, you can access all versions of all files that have been backed up to the device in question. Find the file you want, and just drag and drop it to your desktop. There's no backup GUI to learn and no application to install; just use the tools you already know.

This feature breaks decades of backup tradition by giving you another way to access your backups. For too long our files and databases have been put into backup formats that required the backup software to extract them. We've lived with this for so long that it's actually quite hard to imagine the possibilities that this brings to the table. Here's a short list to help you wrap your brain around this one:

- You can point a full-text search appliance directly at your backups and search the full text of all files ever backed up.

- If you’re running multiple backup products, users and administrators can use a single method of recovery.
- Imagine how easy end-user recoveries would be if you could just point them at a mount point such as *\\backupserver\yourclientname\date*.
- If the disk device allows you to mount the backup read/write, you could actually use the backup as the production filesystem if the production filesystem were down.
- What about using this feature to move between backup products?

These last two ideas deserve more treatment than a single sentence. Suppose you could mount your “backup” as a read-writable volume. Your production database could mount that volume if the primary volume were damaged. Any changes would be stored and could be copied back to the primary system after the primary volume was restored.

Finally, the last bullet point—moving between backup products—is also quite important. Anybody who has been doing backups for a while has been presented with the situation in which you’ve been using backup product XYZ, and you no longer want to do so. Maybe the product is no longer available, or maybe the vendor providing the product has not been innovating to the degree that its competitors have. Historically, moving from one backup product to another meant that you had to keep your old backup software up and running as long as you wanted to restore from your older backups.

What if you had been backing up to a content-aware product? You would no longer need your old backup software to restore from its backups. One option would be to leave the old backups in the old format and just use the web page or filesystem access to restore from them. The

second option would be to mount the old backups to the new backup server and back them up to the new format. Just think of the flexibility that this offers you.

Are Disks Supposed to Be Left Powered Off?

The short answer is no. Disks left powered off were never designed to store data long term. If you ask a disk manufacturer how long you should be able to leave a drive powered off, they will probably tell you it's not a specification that they track. Disk drives are meant to be turned on.

Therefore, there is some question whether removable disk drives are an acceptable medium for off-site backups. Most vendors marketing these products are suggesting that you use them only for "cyclical" backups that are stored for short periods of time, such as a few weeks or months. One vendor is claiming to use something analogous to RAID within a single disk to remove the issues with individual parts of the disk becoming damaged. That vendor uses parity stored on the same disk to allow it to reconstruct data if it is lost due to damaged sections of disk. Time will tell whether these claims are valid.

Stacking

Some integrated VTLs offer an additional feature, known as *stacking*. Stacking is copying multiple virtual tapes onto one physical tape, a feature they borrowed from mainframe virtual tape systems (VTSs). This made sense in mainframes, where applications were unable to append to a tape. The VTS would present to the application hundreds of small virtual tapes and then stack those virtual tapes onto one physical tape, saving tens of thousands of dollars in

media. The value of this feature in most open-systems environments is highly questionable because any decent backup product can append to a tape until it is full. Be aware that the use of this feature breaks the relationship between the backup software's media manager and the physical tape. Also, be aware that some products that support stacking must read the entire stacked tape to read just one of the virtual tapes stacked on that tape. Therefore, you should use this feature only if you are gaining a similar benefit to that achieved in the mainframe environment.

Notification

It is also important to consider which type of *notification* your disk device supports, especially if you are considering an integrated VTL.[1] Some support SNMP traps and some support email notification while others require you to log in to a web page to be notified of any issues.

Disk-As-Tape: Virtual Tape Cartridges

Another way disk emulates tape these days is through virtual tape cartridges (VTCs). VTCs are designed to go inside a tape library and behave like tape in every way—including removability—except they're actually disk. At this writing, this is a brand new concept that may or may not catch on. There are currently three different styles of VTCs.

One VTC uses an array of disks in a canister that's larger than a normal tape but can be handled by a particular model of tape library. It moves the VTC in and out of a virtual tape drive and puts it back into a slot that allows you to eject it like any other cartridge. The advantage to this method is that the VTC is RAID-protected. The

disadvantage to this method is that it is a standard size that works only with one vendor's tape library.

Another VTC puts a single laptop-style drive into a cartridge that looks exactly like an LTO cartridge and uses a specially designed virtual tape drive that accepts only this cartridge. If you put an "LTO" VTC in a real LTO drive, it will eject it, and vice versa. Both the drive and tape have exactly the same form factor as LTO, so they could be integrated into any library vendor that supports LTO. This VTC is, of course, no more or less RAID-protected than another tape. The cost is higher than physical tape, but this type of VTC gives you the benefits of disk and tape in a removable cartridge.

The final style of VTC is from a few companies that didn't feel the need to make a cartridge that was the same form factor as an existing tape. These cartridges are therefore probably going to be limited to standalone use only.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

---

[1] An integrated VTL copies virtual tape to physical tape outside the knowledge of the backup server. If something goes wrong, the only notification you will have will come from the VTL; therefore, notification features are much more important in an integrated VTL.

Part 4. Bare-Metal Recovery

Part IV consists of five chapters:

Chapter 10, Solaris Bare-Metal Recovery

Explains how to use Sun's flash archive to perform bare-metal recovery.

Chapter 11, Linux and Windows

Explains a number of procedures and tools that can be used to perform bare-metal recovery of both Linux and Windows systems.

Chapter 12, HP-UX Bare-Metal Recovery

Covers the `make_net_recovery` and `make_tape_recovery` tools, which now come with HP-UX, to perform bare-metal recoveries.

Chapter 13, AIX Bare-Metal Recovery

Discusses AIX's `mksysb`, probably one of the oldest and best-known bare-metal recovery tools.

Chapter 14, Mac OS X Bare-Metal Recovery

Covers how to perform your own bare-metal recovery of a Mac OS X machine.

Chapter 10. Solaris Bare-Metal Recovery

Flash archive is a flexible, transportable, and easy-to-use utility that can perform installation, cloning, and bare-metal recovery of Solaris systems. Sun documentation presents this product primarily as a cloning and installation tool. However, since you can create an image from a running operating system and then use that image during a boot process to “install” that image to a vanilla system, it makes a perfectly good bare-metal recovery tool, and that is what I focus on in this chapter. It can be likened to the AIX `mksysb` command because it works very similarly.

Flash archive eliminates all the limitations of the Solaris `ufsrestore` utility, which required that all systems have the same hardware, kernel, and device tree set up in order to perform bare-metal recovery. Flash archive is available for Solaris 8 and above, and is fully supported across the full line of Sun servers using either 32- or 64-bit kernels.

Tip ✎

This chapter was contributed by Aaron Gersztoff. Aaron has worked in the enterprise data protection and disaster recovery fields for over 10 years and is an avid baseball fan who has aspirations of visiting every park in the major leagues.

Using Flash Archive

The following section gives you an overview of how flash archive can be used to perform bare-metal recoveries. It also provides a list of setup questions to consider when setting up flash archive.

Backup and Recovery Overview

The following is a quick overview of the basic process of using flash archive to back up and restore the Solaris operating system.

Perform the backup

1. Create a flash archive image on disk (such as an NFS mount) or tape using the `flar create` command.
2. If you create the flash archive on disk, you can also create a tape image from the disk image using `dd`.

> **Tip ✎**
>
> There is also a process for creating a bootable CD or DVD from this image, but that process is not covered in this chapter.

Perform the restore

1. If you plan to restore from tape, place the flash archive tape in a tape drive local to the system being restored.
2. Boot from the system to be restored either from a Solaris Installation CD or via the network.
3. Continue the normal boot process until you reach the Specify Media panel.
4. Select either Network File System or Local Tape, based on where your flash archive image is located.
5. Select the image to restore from, and follow the prompts to complete the restore.

Tip ✎

If you set up a noninteractive restore, steps 3–5 are automatically performed using your configuration files.

Initial Considerations

Before using flash archive, there are many things to consider. This section will help you decide which methods of backup and recovery are best for you, based on your requirements.

System requirements

Flash archive has the following minimum system requirements:

- Solaris 8 (HW 04/01) or later
- Sufficient disk space to store the flash image or a supported tape drive connected either locally or remotely to the system.

Warning ☠

Please test your flash archive recoveries before you really need them. You don’t want to find out that your new system isn’t supported or that your method of recovery won’t work.

When restoring an existing image, your Sun system must have a CD/DVD drive and a bootable CD/DVD of the Solaris operating system, or access to a Solaris network boot

server that supports flash archive installations. The version of Solaris used should be the same or greater than the version of Solaris used on the source system. For example, when recovering a Solaris 9 (HW 09/05) server, you should boot from a Solaris 9 (HW 09/05) or later CD/DVD. (The urgency of this requirement varies depending on the OS version, but I strongly recommend that you use the same or later version CD to avoid potential problems.)

If you plan to perform restores to dissimilar hardware, the system being backed up should include the entire Solaris distribution plus OEM support (*SUNWCXALL*). If you are restoring a system with a limited Solaris distribution to a system with different peripherals or of a different architecture (for example, Sbus versus PCI bus), required drivers may be missing and the recovery can fail. If you plan to perform restores on the same hardware, this is not a requirement.

Frequency of backup

Your individual requirements determine how frequently you create flash archive images. If you're using flash archive images only for bare-metal recovery purposes, you definitely need to create a new image every time you make a change to the server that you want included when performing a bare-metal recovery. Most users of flash archive, though, automate the creation of fresh flash archive images on a regular basis, which makes sure that your system images are always up to date.

Back up to disk or tape?

Flash archive images can be written directly to disk or tape, and you'll need to make a choice as to which option works for you. How you plan to use the images plays a

large part in determining whether to use tape or disk. For example, if flash archive images are created solely for the purpose of off-site bare-metal recovery, you may conclude that a tape-only environment is right for you.

Most environments use a combination of tape and disk by storing a flash archive image on disk, and then copying to tape periodically. This lets you send a copy of the image off-site for disaster recovery purposes and retain a copy on-site to restore a server locally. Companies may also choose to implement a *flash image server*. In this case, flash images are stored on and restored from a central server. Many system administrators also use their existing Jumpstart boot servers to store their flash images (Jumpstart is a Sun tool used to simplify Solaris installation). This provides a network-based boot and an image to recover from all in the same place. In either case, the server may also have one or more tape drives attached to provide an easy way to create portable copies of the image for storage off-site.

If you are considering using disk, keep in mind that sufficient disk space is required to store the flash archive images. Depending on the options selected during the image creation process, a flash archive is typically between 75 and 100 percent of the size of the filesystems included in the archive.

Restore from tape or disk?

A flash archive image can be used locally to restore the Solaris operating system on a server that has failed. In this scenario, either tape or an NFS mount provides an excellent source from which to recover. If hardware and network resources permit, it is recommended that images be stored on disk and restored over an NFS mount. Generally, this method requires less locally attached

hardware (useful when recovering multiple servers) and is usually faster than tape.

A flash archive image can also be used to restore a system in an off-site location, such as in the case of a disaster recovery. In this scenario, a company may decide that restoring critical servers directly from tape is its best option. Others, depending on several factors (number of servers to restore, hardware availability, etc.), may decide to restore all flash images to disk and perform the actual flash recoveries from an NFS mount.

Finally, another option worth mentioning is the use of cloning in conjunction with flash images to build new servers. This option allows users to deploy a standard, hardware-independent build to servers much quicker than if Solaris had to be installed separately on each system. Since the purpose of this chapter is bare-metal recovery, this option is not covered here.

Interactive or noninteractive restore?

When deciding how to set up flash archive, you first have to decide whether you want to be able to perform an interactive or noninteractive restore. An *interactive* restore requires almost no initial setup but more input is needed during the restore process. It may also entail more significant post-recovery efforts. A *noninteractive* restore method requires more setup work but calls for little to no input during the restore process (when time is typically critical).

Either method is considered acceptable. However, since time is usually most critical when a server is down, most system administrators find the time spent creating the files and scripts up front for the noninteractive restore method to be a worthwhile investment.

Other environmental constraints

Many organizations use DMZs and other network tools to restrict access from certain systems to the production network. If this applies to your company, consider how it will impact your implementation of flash archive.

It is recommended that the flash archive infrastructure in a DMZ should mirror what is available in production. Examples of required infrastructure include tape drives, NFS servers, and backup servers. If resources are not available to mirror the production infrastructure, be sure to have a tested, reliable process in place to back up and restore the servers in the DMZ.

Preparing for an Interactive Restore

The easiest way to get started using flash archive is to create backups for an interactive restore. This requires more work during the recovery process, but you'll have a bootable backup in only a few minutes. Later in this chpater, we'll explain how to take this backup to the next level by making the restore proceed without interaction.

Creating Flash Archive Images

Building a solid, repeatable, and tested backup process is the key to being able to restore when you need to. This section describes the steps required to create a flash archive image, from determining which filesystems to include to the tape creation process (if desired).

Determining filesystems to back up

While any filesystem can be included in a flash archive, typically, only OS filesystems are required to restore a Solaris system. Data restored from filesystems not created during the recovery process is likely to be restored to the root filesystem. This could have serious consequences, especially if the total amount of data to restore exceeds the size of the root filesystem.

> **Tip ✎**
>
> If you plan on performing a noninteractive restore, and you decide to include any non-OS filesystems in the backup, you need to include those filesystems in your *profile* file if you would like them created prior to the restore. (Noninteractive setup is covered later in this chapter.)

Using flar create

`flar create` is the Solaris utility that creates flash images. The process can either be scripted and run through a scheduler, such as `cron`, or run manually. If you run `flar create` manually and you plan to perform a noninteractive restore, the image should be created on disk and then copied to tape, due to prerequisites in the tape creation process.

The `flar` manpages provide current syntax and usage of the command. Here are the options used in the examples:

`create`

Tells `flar` to create an archive.

`info`

Without any additional flags, `info` examines an archive and displays the summary information it finds in the archive. If given the additional `-l` flag, it displays the files found in the archive. (`create` and `info` are mutually exclusive options.)

`-c`

Tells `flar` to compress the archive as it’s writing it.

`-n`

Gives the archive a name that is stored inside the archive. We can query this name later in case the filename we stored it under isn't very helpful.

`filename` or *`tape_device`*

Passes `flar` the name of a filename or tape device to write to.

Here is an example of the `flar create` process running on a Solaris 9 system. In this example, `flar` compresses the archive (`-c`), gives it a name of *sun2.flar* (`-n sun2.flar`), and uses *sun2.flar* as the filename to back up to as well:

```
# flar create -c -n sun2.flar sun2.flar
Full Flash
Checking integrity...
Integrity OK.
Running precreation scripts...

Precreation scripts done.
Determining the size of the archive...
3949130 blocks
The archive will be approximately 1.01GB.


Creating the archive...
3949130 blocks
Archive creation complete.
# echo $?
0
```

Note that at the end of the archive, we checked the return code by entering `echo $?`. The Solaris return code is the primary method of determining the result of the `flar create` command. A return code of zero indicates a successful completion while any nonzero return code indicates a failure.

After receiving a zero return code from the `flar create` command, you can use the `flar info` and `flar info -l`

commands to ensure that the image creation process was successful. If the return code of `flar create` or the output of either `flar info` command indicates a problem, the archive should be considered suspect. The following example shows an example of the `flar info` *`imagename`* command:

```
# flar info sun2.flar
archive_id=c80af5b0c18ef7a9375e124c07f56875
files_archived_method=cpio
creation_date=20060211192433
creation_master=sun2
content_name=sun2.flar
creation_node=sun2
creation_hardware_class=sun4m
creation_platform=SUNW,SPARCstation-5
creation_processor=sparc
creation_release=5.9
creation_os_name=SunOS
creation_os_version=Generic_118558-11
files_compressed_method=compress
files_archived_size=470522936
content_architectures=sun4m
type=FULL
```

In addition to displaying general information about the flash archive image, you can actually display a list of files included in the image using the `flar info –l` *`imagename`* command:

```
# flar info -l sun2.flar
lost+found
export
export/home0
export/home0/lost+found
export/home0/rules
export/home0/rules.ok
export/home0/sysidcfg
export/home0/standard.profile
~list truncated~
platform/sun4u/kernel/drv/sparcv9
platform/sun4u/kernel/drv/sparcv9/scmi2c
```

Creating a flash archive tape

Once the disk image has been created, you can create a tape to perform a bare-metal recovery. All you have to do is use the dd command:

```
# mt –f /dev/rmt/0n rewind
#dd if=sun2.flar of=/dev/rmt/0n obs=1024000
```

Tip ✎

When creating a flash tape, use a tape drive that uses native Solaris OS drivers and does not use nonstandard entries in the */kernel/drv/st.conf* file. Using nonstandard entries in the *st.conf* file may cause problems not noticed until performing a restore.

Bare-Metal Recovery with Flash Archive

Now that we've created a flash archive image, let's use it to perform an interactive restore of a Solaris system. To do this, perform the following steps:

1. First, ensure that the system is powered on and that it is at the ok> prompt.
2. If you are unsure whether your boot and recovery devices are available, you can ensure they are by performing a probe:
3. ```
   ok> probe-scsi-all
   ```
4. A list of locally attached devices should be displayed. Ensure that at the minimum, the local boot disk, a CD/DVD drive (if applicable), and the correct SCSI tape drives are listed.
5. If you plan to restore from tape, insert the tape into a local tape drive.
   ```

6. Boot the system:
7. a. If you are booting from CD, insert the Solaris Installation CD into the appropriate device on the system you want to restore and enter **`boot cdrom`** at the `ok>` prompt.

   b. If you have a network boot server configured, you may also boot from the network using the `boot net – install` command.
8. The system then boots, and you are taken through the standard Solaris installation prompts. Once you reach the Specify Media prompt, select `5` to restore from local tape, or `2` to restore from an NFS mount.
9. a. If you select tape, you are prompted for the tape device name and the tape file in which the image is located. Remember that `0` is the first file on the tape.

   b. If you select NFS, you are prompted for the name of the NFS server and share where your flash archive images are located. You will then need to select an image from the images it finds there.
10. You are then prompted for the name of the disk you wish to restore to and asked how you would like it partitioned. Answer the prompts according to your environment.
11. Once those steps are done, the recovery should complete automatically, followed by a reboot.

An interactive recovery example

The following example restores an image from tape. If your images are on disk, you would select option `2` instead of option `5`.

```
Please specify the media from which you will install the Solaris
Operating
Environment.

Media:

1. CD/DVD
2. Network File System
3. HTTP (Flash archive only)
4. FTP (Flash archive only)
5. Local Tape (Flash archive only)

   Media [1]: 5

Please specify the local tape device and the position on the tape
where the
Flash archive is located.

To select a different media, enter B to go Back.
   Tape Device or B [/dev/rmt/0] /dev/rmt/0n
   Tape Position or B [1] 0

You selected the following Flash archives for initial install:

 Tape  /dev/rmt/0n      0

Press Return to continue or enter D to deselect all archives:
<enter>

To select additional Flash archives, please specify the media
where the
archives are located.

Media:

1. CD/DVD
2. Network File System
3. HTTP
4. FTP
5. Local Tape
6. None - Archive Selection Complete
```

If there are archives that contain additional filesystems, you can specify them here.

```
   Media [6]: <enter>

The system is being initialized, please wait... -
```

```
Select which disks you want to lay out the file systems on.
Available Disks:
   Disk      Size
 c0t0d0    4000 MB

Enter 'y' to layout file systems on the specified disk.  This
will erase all
existing data on the disk.  Enter 'n' to leave the disk
unmodified.  Enter 'e'
to leave the remaining disks unmodified and continue with
install.

Layout file systems on disk c0t0d0 (bootdisk) (y/n) [y]? y

At least one of the disks selected for installing Solaris
software has file
systems or unnamed slices that you can choose to preserve. Do you
want to
preserve existing data?

Enter y to preserve data or n to skip data preservation. [n]
<enter>

The Solaris Installer is determining size requirements based on
your choices,

please wait...  \

File System operations:

1. Print the current partition table
2. Modify a disk's partition table
3. Return to beginning
4. Done

Select the number corresponding to a file system operation,
'Return to
beginning' to change selections, or 'Done' to proceed with the
install [4]: 1

Disk Name  Slice Name    Size(Mb)
c0t0d0
            /             2000
            swap          514

File System operations:

1. Print the current partition table
```

```
2. Modify a disk's partition table
3. Return to beginning
4. Done

Select the number corresponding to a file system operation,
'Return to
beginning' to change selections, or 'Done' to proceed with the
install [4]: 4

The following items will be installed:

Tape  /dev/rmt/0n      0
Root  Device: c0t0d0
File  Systems:
      c0t0d0s0 / 2000 MB
      c0t0d0s1 swap 514 MB

Enter 'y' to accept these values and start the installation, or
'n' to
return to disk selection to make changes (y/n): y

Installing...

Extracting archive(s)
Enter 'y' to accept these values and start the installation, or
'n' to
return to disk selection to make changes (y/n): y

Installing...   (Note:  The progress meter may not move during the
extraction.)

Extracting archive(s)
|-1%--------------25%-----------------50%-----------------75%----
----------100%|

...lines deleted...

Feb 18 09:19:45 rpcbind: rpcbind terminating on signal.
syncing file systems... done
rebooting...
Resetting ...
```

The flash recovery is now complete. Once the system reboots, you may have to configure system information such as the node name and network information.

Setup of a Noninteractive Restore

Setting up a noninteractive restore entails more work, but it can help quite a bit during recovery. In addition to allowing the recovery to complete without interaction, a noninteractive restore can also add the following features:

- Setup of preinstallation tasks and configuration of variables prior to restoration for systems with dissimilar hardware
- Post-installation tasks prior to the final reboot, such as software installation, boot process modifications, etc.

Noninteractive Setup Files

Jumpstart has been used for many years to automate the installation of Solaris. Flash archive uses some of the same technology as Jumpstart; for example, it uses the concept of the *profile*, *rules*, and *sysidcfg* files to provide for a noninteractive restore from a flash archive image. Therefore, in addition to creating your flash archive, you need to create these files to prepare for a noninteractive restore. After creating these files, ensure that root is the owner of each file and that permission on each is set to 644.

profile

A *profile* file specifies the filesystems to be created, the disks they are created on, and the install method used during the recovery. The profile can be named anything as long as the name of the profile is specified in the *rules* file. Profiles can be static or created dynamically via a *begin script*. Profiles that are created dynamically by a begin script are called *derived profiles*.

Here is an example of a static profile. A static profile contains several rows of parameters followed by values, separated by whitespace. When creating a static profile, ensure root owns the file and that permissions on the profile are set to 644. The

following parameters should be included in every profile created for use with flash archive:

`install_type`

This should say `flash_install` to indicate this is a flash recovery. (Remember profiles are used for other things, such as Jumpstart.)

`archive_location`

Specify where this unattended restore gets its data from. This can be either `local_tape` *`device_name file_#`* or `nfs` *`nfserver:/path/imagefile`*.

`partitioning`

Since we are restoring from a flash archive, use explicit partitioning so that you can create filesystems that support your flash recovery as well as your overall recovery plan. When using the `explicit` option, use the `filesys` keyword to define and partition available disk space.

`filesys`

Specify one or more lines containing the names of the filesystems (and swap) partitions. Each line must contain the keywords `filesys` *`diskname size mountpoint`*:

```
install_type    flash_install
archive_location        local_tape      /dev/rmt/0n 1
partitioning    explicit
filesys c0t0d0s0        7000 /
filesys c0t0d0s1        2000 swap
```

While this example shows an existing static profile that is copied to tape, a begin script that can create a derived or custom profile on the fly during the recovery process provides increased flexibility when restoring to systems with dissimilar hardware. Examples of this flexibility include probing system devices, prompting for user input when selecting and partitioning disks, and selecting which tape drive will be used.

Here is a basic begin script that creates a derived profile that looks like the static profile shown later in this chapter. Like the

profile, this script can be named anything; it just has to be named in the *rules* file.

```
#!/bin/sh
echo "install_type    flash_install"                              >
${SI_PROFILE}
echo "archive_location         local tape    /dev/rmt/0n 1" >>
${SI_PROFILE}
echo "partitioning     explicit"                                    >>
${SI_PROFILE}
echo "filesys       c0t0d0s0        7000    / "                 >>
${SI_PROFILE}
echo "filesys       c0t0d0s1        2000    swap"              >>
${SI_PROFILE}
```

Tip ✎

When creating a begin script, you must use the `${SI_PROFILE}` variable as shown here.

Again, this is very basic, and there wouldn't be much point to creating a script like this one because it creates a profile just like our static example. However, with a little creative scripting, you can turn a begin script into a very useful tool that can probe system devices, check minimum disk sizes, and request user input for unknown variables. For example, it can display a list of tape drive devices and ask the user to specify which one she wants to use. More information on creating a begin script can be found in the flash archive recovery documentation.

The rules file

The *rules* file specifies which profile, begin scripts, and finish scripts are used during the restore process. A *rules.ok* file is created automatically after running the `check` command against your *rules* file. The layout of this file is as follows:

```
[keyword(s)]    [keyword(s) value]     [begin script]  [profile]
[finish script]
```

For example:

```
any             -             -        standard        -
```

This example states that any (or every) system should use the standard profile to configure the restored system, with no begin or finish scripts. Jumpstart *rules* files can be a lot more complex and powerful. While a *rules* file requires only one rule, you can specify many rules for setting up systems based on configuration, architecture, and more.

The sysidcfg file

The *sysidcfg* file contains information used to configure the system during the restore process. Though not shown in the following example (and generally not recommended from a security perspective), an encrypted root password can also be included in the *sysidcfg* file. For more information on using this option, refer to the Solaris Installation documentation.

Here is an example of a *sysidcfg* file. The keywords and their meanings are relatively straightforward.

```
system_locale=C
timezone=US/Pacific
network_interface=primary {hostname=sun2
        default_route=192.168.1.253
        ip_address=192.168.1.34
        netmask=255.255.255.0
            protocol_ipv6=no}
security_policy=NONE
name_service=NONE
timeserver=localhost
```

Once these setup files are created, you can integrate them with the flash archive image created in the section "Preparing for an Interactive Restore," earlier in this chapter.

Creating a Noninteractive Tape Image

Once the image has been created, you can create a tape that can be used to perform a bare-metal recovery. The following steps create a flash archive image on disk and copy it to tape:

1. Create *sysidcfg*, *rules*, and *profile* (or *begin script*) files.
2. Run `check` against the *rules* file to create *rules.ok*.

3. Create flash image on disk, using `flar create` (see the earlier section “Preparing for an Interactive Restore”).

> **Tip ✎**
>
> If the `check` script is not on your local server, it can be run from or copied from the Solaris CD/DVD.

1. `tar` *sysidcfg*, *rules*, *rules.ok*, and *profile* (or *begin script*) files to the tape.
2. `dd` the flash image to tape. Alternatively, you can follow steps 1–4, then create the flash archive image directly to tape using the `flar create` command.

The following is an example of this process starting with step 4:

```
# tar cvf /dev/rmt/0n rules rules.ok sysidcfg standard.profile
a rules 1 tape blocks
a rules.ok 1 tape blocks
a sysidcfg 1 tape blocks
a standard.profile 1 tape blocks
#dd if=sun2.flar of=/dev/rmt/0n obs=1024000
```

Creating a Noninteractive Disk Image

If you wish to perform a noninteractive restore from an NFS mount, you should have a properly configured Solaris network boot server configured on your network and boot using the `boot net – install` command. In this case, how your individual environment is set up dictates where the server configuration files (*profile*, *rules*, and *sysidcfg*) should be stored.

Building and booting from a custom Solaris CD/DVD that restores using an existing Jumpstart server over the network is also possible but is not covered in depth in this chapter.

To perform a noninteractive restore:

1. First, ensure that the system is powered on and that it is at the `ok>` prompt.

2. Check whether your boot and recovery devices are available by performing a probe:
3. `ok> probe-scsi-all`
4. A list of locally attached devices should be displayed. Ensure that, at a minimum, the local boot disk, a CD/DVD drive (if applicable), and the correct SCSI tape drives are listed.
5. If you plan to restore from tape, insert the tape into a local tape drive.

Tip ✎

In the example *profile* file, there is one tape drive connected to the server (*/dev/rmt/0n*). If a tape drive is hardcoded into the *profile* file, that same drive must be used here. To avoid this requirement, use a begin script to create a profile, on the fly, during the restore process.

1. Boot the system.
2. a. If you are restoring from tape, insert the Solaris Installation CD into the appropriate device on the system you want to restore. At the `ok>` prompt, enter:

   b. `boot cdrom - install tape=/dev/rmt/0n`

   c. The system boots, and the installation process starts. The system boots from the CD/DVD and looks for the required files on tape. If present, any prerecovery scripts execute, and the archive extraction begins. The extracted *rules.ok*, *profile*, and *sysidcfg* files are used to configure the server.

   d. If you are restoring from an NFS mount, you need to have a network boot server configured and boot from the network using the `boot net – install` command. The system looks for the network boot server to provide

the location of the required configuration files and flash image.

Here's the output from a noninteractive restore:

```
ok boot cdrom - install tape=/dev/rmt/0n
Resetting ...

SPARCstation 5, No Keyboard
ROM Rev. 2.15, 96 MB memory installed, Serial #7760400.
Ethernet address 8:0:20:76:6a:22, Host ID: 80766a00.

Rebooting with command: cdrom - install tape=/dev/rmt/0n
Boot device: /iommu/sbus/espdma@5,8400000/esp@5,8800000/sd@6,0:d
File and args: - install tape=/dev/rmt/0n
SunOS Release 5.9 Version Generic_118558-11 32-bit
Copyright 1983-2003 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Configuring /dev and /devices
Using RPC Bootparams for network configuration information.
Skipping interface le0
Skipping interface hme0
Searching for configuration file(s)...
Using sysid configuration file from /dev/rmt/0n position 0
Search complete.
The system is coming up. Please wait.
Begin system identification...
Starting remote procedure call (RPC) services: sysidns done.
System identification complete.
Generating software table of contents [this may take a few minutes...]
Table of contents complete.
Starting Solaris installation program...
Searching for JumpStart directory...
Using rules.ok from tape.
Checking rules.ok file...
Using profile: standard.profile
Executing JumpStart preinstall phase...
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.

Processing profile
        - Opening Flash archive
        - Validating Flash archive
        - Selecting all disks
        - Configuring boot device
```

```
        - Configuring / (c0t1d0s0)
        - Configuring swap (c0t1d0s1)
        - Configuring /export/home (c0t3d0s7)

Verifying disk configuration
        - WARNING: Unused disk space (c0t1d0)
        - WARNING: Unused disk space (c0t3d0)
Verifying space allocation
        NOTE: 1 archives did not include size information
Preparing system for Flash install

Configuring disk (c0t1d0)
        - Creating Solaris disk label (VTOC)
Configuring disk (c0t3d0)
        - Creating Solaris disk label (VTOC)
Creating and checking UFS file systems
        - Creating / (c0t1d0s0)
        - Creating /export/home (c0t3d0s7)

Beginning Flash archive processing

Predeployment processing
8 blocks
16 blocks
8 blocks

No local customization defined

Extracting archive: sun2.flar
        Extracted    0.00 MB (  0% of  448.73 MB archive)
        Extracted    1.95 MB (  0% of  448.73 MB archive)
               ~truncated~
        Extracted  448.72 MB ( 99% of  448.73 MB archive)
        Extraction complete
Postdeployment processing
No local customization defined
Customizing system files
        - Mount points table (/etc/vfstab)
        - Network host addresses (/etc/hosts)
Cleaning devices
Customizing system devices
        - Physical devices (/devices)
        - Logical devices (/dev)
Installing boot information
        - Installing boot blocks (c0t1d0s0)
Installation log location
        - /a/var/sadm/system/logs/install_log (before reboot)
        - /var/sadm/system/logs/install_log (after reboot)
Flash installation complete
Executing JumpStart postinstall phase...
The begin script log 'begin.log'
```

```
is located in /var/sadm/system/logs after reboot.
syncing file systems... done
rebooting...
Resetting ...
```

Once the restore completes successfully, any post-recovery scripts are then executed, and the system reboots. If you did not specify an encrypted root password in the *sysidcfg* file, you are prompted to enter a root password during the next boot. The recovery is now complete.

Post-Recovery Procedures

Regardless of the method you use (interactive or noninteractive), once the flash archive has been restored, some post recovery work may need to be completed. How much and how complex that work is varies depending on the system configuration. It is very important to understand how your systems are set up to allow for the building of a successful end-to-end recovery procedure.

For example, if a system with a VxVM encapsulated root disk has been restored, the system will not boot following the flash recovery. It doesn’t boot because Veritas Volume Manager is expecting to find an encapsulated root disk, which no longer exists. In this scenario, you must boot from the Solaris CD into single-user mode and make the necessary changes to disable the loading of VxVM for the system to boot properly.

If you restore to a system with a different root disk, you may need to reconfigure the boot device *EEPROM*:

```
Original Server Root Disk        Target Server Root Disk

c0t0d0s0                                 c1t0d0s0
```

Final Thoughts

Pre- and post-processing can be scripted and included in the flash archive process. Think about activities you can automate within your recovery or build processes. Here are some common pre-processing examples:

- The ability to interactively choose which disks, slices, and sizes are used during the recovery process
- Using a begin script to create a system *profile* file during the recovery process

Common post-processing examples include:

- Non-OS filesystem layout
- Modifications to the boot process
- Making changes to the SUN *EEPROM*

Flash archive is a significant improvement over traditional methods of building and restoring Solaris systems. It removes critical dependencies long considered acceptable and gives the user more flexibility than ever.

With that said, flash archive, much like any other program, utility, or script, is useless without clear, accurate, and tested processes and procedures. Any tool is only a part of an overall process that should be documented and tested periodically to maximize the potential for success.

There are many great sources of documentation on the use of flash archive. A couple of very helpful resources are the Solaris installation and advanced installation guides, available at http://docs.sun.com, and the Sun BluePrints at http://www.sun.com/blueprints.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 11. Linux and Windows

This chapter explains how to perform a bare-metal recovery of Linux or Windows running on a standard Intel system. You should be able to use this procedure to recover any other operating system that runs on standard Intel systems. It has been tested with CentOS 4.0.2 (also known as RedHat Enterprise 4), Windows 2000, and Windows XP. (This procedure does *not* work with Macintosh Intel systems because they are customized for Mac OS. See Chapter 14 for the Mac OS procedure.) The procedure requires only open-source tools.

> **Tip ✎**
>
> This chapter is a collaboration between Reed Robins and W. Curtis Preston. Reed is a data protection specialist with GlassHouse Technologies.

This procedure works for a Windows-only system, but it does use a Linux distribution that can run straight from a CD. Windows users will have to run a few Linux commands, but we do our best to keep them simple. If we had found an open-source bare-metal recovery tool that runs on Windows, we would have written a separate chapter for it. Unfortunately, at this writing, that doesn't exist.

If you're a Windows user who has never touched Linux, we urge you to at least try the procedures in this chapter before giving up on the idea. If you still find them too difficult, we also cover G4L, which is an open-source product that is similar to the commercial product Ghost. It uses Linux but requires much less interaction with the operating system. Finally, we also mention a few commercial products at the end of the chapter.

Due to the availability of inexpensive disk and the ease of recovery that disk brings to the table, this is an entirely disk-based procedure. You can use a Unix NFS share or a Windows share as a target. Any disk-like target that can be mounted before a backup or restore will work with this procedure. If you're interested in other disk targets, it would be worthwhile to read through the complete backup and recovery HOWTO on the Linux Documentation Project at http://www.tldp.org/HOWTO. This HOWTO has helpful hints for reducing the amount of information being backed up for a minimal OS restore.

How It Works

Most system administrators are familiar with the typical bare-metal recovery answer: install a minimal operating system and recover on top of it. However, this procedure presents several problems. It takes too long, you could run into open-file conflicts, and it is difficult to document and recover operating system customizations.

Like the other bare-metal recovery procedures in this book, this procedure does not require a reinstallation of the operating system in order to recover it. It breaks down into six major steps:

1. Back up the important metadata.
2. Back up the operating system with a native utility.
3. Boot from alternate media.
4. Prepare the new root drive.
5. Restore the boot block to the new root disk.
6. Restore the operating system information.

We'll start with some background and then describe the general aspects of each step. Ultimately, we will provide a detailed procedure for each method described. Before you can choose a backup and recovery method, though, you have some decisions to make, as outlined in the next section.

Booting from Alternative Media

In order to easily recover your operating system, you need a limited root shell (also known as a *mini-root*) where you can run `fdisk`, `mkfs`, `gzip`, `pax`/`tar`/`cpio`, `dd`, `ntsfclone`, and so on. You also need support for whatever backup media you have chosen. This functionality is provided via rescue floppies or a bootable Linux distribution on CD, also known as a LiveCD. Rescue floppies, such as *TomsRtBt*, contain a minimal version of Linux and usually contain the recovery utilities listed previously. They also have several drivers for popular backup media, such as parallel-port ZIP drives, SCSI tape drives, NFS, and SMB/CIFS. LiveCDs are more complete versions of Linux and come in a variety of flavors. They typically have support for more drivers and utilities than the rescue floppies. This expanded support and flexibility combined with the now ubiquitous nature of CD-Rs is what caused us to select a LiveCD for our examples.

A list of Intel rescue floppies may be found at http://metalab.unc.edu/pub/Linux/system/recovery/index.html. A list of LiveCDs may be found at http://www.frozentech.com/content/livecd.php. The Linux LiveCD that we chose to use is Knoppix (http://www.knoppix.org).

If Then GOTO

This chapter spends the next several pages explaining a lot of the theory and manual steps behind a bare-metal recovery of an Intel system. Just like a lot of manuals, the good stuff is at the end. If you want to jump right to the method that applies to you, this section should give you just enough detail to know which section to jump to.

If you're running Windows and have never typed anything at a Linux prompt, you should probably proceed to the section "Automate Bare-Metal Recovery with G4L." It describes a menu-driven method of bare-metal recovery. (If you want a menu-driven method, this is the only one we discuss.)

If any of the following apply to you, you must use either G4L or the alt-boot full image method:

- If you're running the Linux Volume Manager (LVM) or other software RAID system
- If you're using an IA64 system
- If your partition tables contain references to an extended partition table

If any of the previous conditions apply to you, the following conditions do not apply to you:

- For a manual method, see the section "Alt-Boot Full Image Method"; for an automated method, check out the "Automate Bare-Metal Recovery with G4L" section.
- If you're running Linux and want to try to create live bare-metal backups, you should jump right to the section "Live Method" later in this chapter.
- If you're using a dual-boot Linux/Windows system, and you don't mind a little downtime to get a bare-metal backup, you should probably check out the alt-boot filesystem or alt-boot partition image method. These

methods can also be automated using the tool discussed in the section "Automate Bare-Metal Recovery with G4L," later in this chapter.

Choosing Backup Methods

Our goal was to perform a bare-metal backup of an Intel system without using commercial tools. To use the procedures we developed, you have to make three decisions:

- Will you back up while the system is running (live) or while it is booted from an alternate boot disk?
- If using an alternate boot disk, will you back up at the block/image level or the filesystem level?
- If backing up at the image level, will you back up the entire drive as one image or as separate partitions?

Partition Your OS Drive

Many people have one large partition for their operating system, applications, and data. Partitioning your hard drive in this manner limits your bare-metal recovery options. A better idea would be to partition your hard drive with one partition just big enough to hold your operating system and its applications and a second partition to hold all your data. Having partitions like this also allows you to back up the data drive using standard filesystem backup techniques (such as Amanda, BackupPC, Bacula, `rsnapshot`, and `rdiff-backup`), and use bare-metal recovery procedures only for the operating system partition.

If this idea interests you, you can use tools such as Partition Magic in Windows or QTParted for Linux to

repartition your drive with multiple partitions. (If you haven't seen QTParted, it's great; it works almost exactly like Partition Magic, but it's Linux-based and free!)

Live or alternate boot?

The first choice to make is whether you will back up your system while it's running (live) or back it up while it's booted from alternate media. Backing it up live involves using filesystem-level backup commands to back up the files on your running system.

If you're running Windows, this choice is already made for you. You need to use the alternate boot method because, as of this writing, there is no backup format that can be written in Windows and read in Linux that also supports ACLs. There are rays of hope, though. We can now easily mount and create NTFS filesystems in Linux, the `mtftar` command can read `NTBACKUP` tapes, and the community developing the `pax` utility says that `pax` is starting to support ACLs.

The biggest advantage to backing up your system live is that it does not require downtime for the backup. Depending on your operating system, though, there may be issues with open files or system configuration databases. If it's important that you back up your system live, just make sure you test the procedure well to make sure that you've dealt with all the issues specific to your operating system and platform.

> **Tip ✎**
>
> If you cannot take your system down for an occasional bare-metal backup, and you can't make the live backup method work for you, read the section "Commercial Solutions" at the end of this chapter.

If you cannot take your system down for an occasional bare-metal backup, the other option is to back up your system while it is booted from an alternate boot disk, such as a LiveCD Linux distribution, during backup. (A LiveCD is a Linux distribution designed to give you a fully functional Linux operating system by simply booting from a CD.) The alt-boot method, as this is called, also offers a number of advantages, such as not having to worry about filesystem formats. It does this by backing up at the block (or image) level, which is not possible when backing up live. It's only disadvantage is that it requires your system to be unavailable during the entire backup. This shouldn't be a problem for home users and some small businesses, but others may find this limitation unacceptable.

> **Tip ✎**
>
> We use the terms *live* and *alt-boot* as shorthand for these two backup options.

Image level or filesystem level?

If you're using the alt-boot method, you can access your data as filesystems or raw disk devices. If you back up the data using the raw disk device (for example, */dev/hda* or */dev/hda1*), we say that you're backing up at the image level. The other option is to back up the data as filesystems by

mounting them and using a tool such as `tar`. (You cannot perform a filesystem backup and restore of NTFS partitions.)

> **Tip ✎**
>
> We'll use the terms *image* or *filesystem* as shorthand for these two backup options.

Backing up at the image level gives you the advantage of not having to care about the operating system that's using the drive. As long as you back up all the bytes that are on the hard drive and restore them back to the same partition, everything will work fine. This is how we can easily back up a Windows system using Linux.

The biggest disadvantage of image-level backups is that they back up every byte on the drive whether it's being used or not. If you've got a 10 GB partition, you're going to get a 10 GB backup, even if there's only 1 GB of files on that partition. (Compression should help get rid of those empty blocks but will probably lengthen the backup time.) Another disadvantage of image-level backups is that they are all-or-nothing. You cannot restore individual files from an image-level backup.

Complete disk or separate partitions?

If you're using the alt-boot method and have decided to back up at the image level, you have another decision to make. Will you back up the operating system drive as one large image (*/dev/hda*) or as separate partitions (*/dev/hda1*, */dev/hda2*, and so on)? In order to back up at the partition level, you need to have multiple partitions on your hard drive. Read the sidebar "Partition Your OS Drive" later in this chapter.

> **Tip ✎**
>
> We'll use the terms *full* or *partition* as shorthand terms for these two backup options.

Backing up the entire OS drive as one large partition has one major advantage: recovery is incredibly simple. You don't have to worry about repartitioning the hard drive for recovery, and you don't have to worry about the boot block (the master boot record, or MBR). Just back up the entire drive as one large image, and you're done.

The disadvantage of this method is the size of today's hard drives. If you back up the entire hard drive as one image, you may need to create a single image that's hundreds of gigabytes—or even a terabyte. Wow! Besides the space required to store such an image, backing up hundreds of gigabytes with one backup command could take an extremely long time—and your system is down the entire time.

The other method would be to create multiple partitions and back up each partition separately. This allows you to use the bare-metal procedure only for the partitions that contain the operating system, and use filesystem backup techniques (such as Amanda, BackupPC, Bacula, `rsnapshot`, and `rdiff-backup`) for the rest of the system. It also allows you to run backups of all partitions simultaneously, speeding up the whole process.

There are disadvantages to the partition method when compared to the full disk method, all of which are about complexity. You'll need to understand more about how your drive is partitioned, and you'll need to worry about backing up and recovering the partition table and MBR. When you partition your hard drive for recovery, you have to create

partitions of the right size to recover to. If you make them too small, the recovery won't work; if you make them too large, you'll be wasting disk.

Four backup options

The three decisions just described translate into four backup methods. Using the shorthand terms described earlier, those methods are alt-boot full image, alt-boot partition image, alt-boot filesystem, and live.

Alt-boot full image

: This method consists of booting to a LiveCD and creating an image of the *entire OS disk*. This is the simplest of all the tasks and the one that works for the most people, but it requires enough downtime to back up the entire OS drive—and enough space to hold that backup.

Alt-boot partition image

: This is similar to the alt-boot full image method, but you make an image of just the partitions you need to back up. This method can save a lot of time and space if your hard drive is properly partitioned, but it requires downtime and a more extensive knowledge of your hard drive contents.

Alt-boot filesystem

: This method consists of booting to a LiveCD such as Knoppix and then using filesystem-aware utilities to back up the hard drives. `tar` and `cpio` are available in Knoppix. This can save you a lot of space if your filesystems aren't very full, but it's the most complicated of all the tasks and still requires downtime.

Tip ✎

If you're going to use the alt-boot filesystem or live backup method, you need to use utilities that are available in the LiveCD that you're using. We've chosen Knoppix because of its popularity and great device support. Knoppix also gives you a wide variety of utilities: `tar`, `cpio`, `dd`, and `ntfsclone` are all available. (`ntfsclone`, also available in Knoppix, is really an image utility, but it is NTFS filesystem-aware, so it enables you to back up only the part of the image that contains files.)

Live

If you're running Linux and are *not* running LVM, software RAID, or using an IA64, you can back up the operating system while it is being used, then use alternate boot media just for the restore. You can use whichever filesystem level backup utility you wish. Remember that this method does not easily support dual-boot systems because they don't have a common filesystem. It also won't support Windows-only users because the commands that they use to back up their system won't be available from the media used for restore. However, if you can't take your system down for an occasional OS-level backup, this may be your preferred method. (With a dual-boot system, you can back up your Windows partitions while you're booted into Linux, of course.)

Tip ✎

All the examples in this chapter use the directory */backups* as the backup target. This can be an NFS- or CIFS-mount or a local removable hard drive. The drive must be writable as root, so add the option `-o rw,root` to any NFS shares.

The Steps in Theory

Later in this chapter, you'll find specific procedures for each of the four methods we've described. This next section takes a closer look at the six general steps involved in our bare-metal backup and recovery process.

Step 1: Back Up Important Metadata

The first data that you need to save is the metadata. Metadata is the data about that data, that is, the information about how the system is physically configured. In this case, it's how the operating system disk is partitioned. This information is not typically included in a normal backup. Making a copy of the MBR and partition table is a way to maintain this information in a format that can be used to recreate the root disk partition.

The MBR and partition table are contained in the first 512 bytes of your hard drive. An MBR has three parts: the boot block is stored in the first 446 bytes, the paritition table is stored in the next 64 bytes, and the boot code signature takes up the remaining 2 bytes.

Linux, FreeBSD, and Solaris x86

If you're running Linux, FreeBSD, Solaris x86, or any other Unix-like operating system, you can easily back up your important metadata live.

Tip ✎

If you are using Linux LVM, software RAID, extended partitions, or IA64 systems, you need to use the alt-boot full image method, which allows you to skip this step.

To save the disk partition information, run these commands:

```
# fdisk -l >/backups/fdisk.txt
#cp /etc/fstab /backups/fstab.txt
```

To back up the MBR and partition table, run the following command. It creates a backup to a file called */backups/mbr*.

```
# dd if=/dev/hda of=/backups/mbr bs=512 count=1
```

This can be used later to restore the MBR and the partition table.

Windows

Unfortunately, there are no Windows equivalents to the commands just shown. You have two choices. If you use the alt-boot full image method, you can skip this step. If you want to use the alt-boot partition image method, save this information before backing up your partitions.

After booting into Knoppix, you are automatically logged in as user `knoppix`. This example assumes you are booting into a DHCP environment and have an NFS server called *nfsserver* with a share called */data08/curtis*. Obviously, you need to substitute the appropriate values for your environment. In addition, if you're using a USB drive for this procedure, you need to mount it instead of the NFS drive. Run these commands to proceed:

```
$ su -
#mkdir /backups
```

> **Tip ✎**
>
> While Knoppix is booted from read-only media, it's an entire OS running from RAM, so you can indeed make directories or even install software into this RAM environment. Of course, everything goes away when you reboot.

Now run one of the following two commands depending on whether you are running NFS or SAMBA:

```
# mount nfsserver:/data08/curtis /backups
# mount -t smbfs -o username=administrator,password=PASSWORD//servername/share
/backups
```

Finally, save the partition information:

```
# fdisk -l >/backups/fdisk.txt
#dd if=/dev/hda of=/backups/mbr bs=512 count=1
```

> **Warning ☠**
>
> This procedure assumes you are not using Windows dynamic disks for your operating system drive.

Step 2: Back Up the OS with a Native Utility

If you are going to restore the operating system without reinstalling it, you need to use a utility that is always available.

Linux, FreeBSD, Solaris X86

If you're using the live method, you'll want to use `tar` or `cpio`. `tar` is easily the most popular choice here because it's more portable than `cpio`. If you're going to use the alt-

boot filesystem method, you need to understand the filesystem types, mount the filesystems, then use `tar` or `cpio` to back them up. If you're going to use the alt-boot full image or alt-boot partition image methods, you need to use `dd`.

Windows

If you're going to back up a Windows system using the alt-boot full image or alt-boot partition image methods, you'll need to use `dd`. We'll cover the syntax that you'll need to know. If you're going to use the alt-boot filesystem method, you'll need to use the `ntfsclone` utility that's available on the Knoppix CD. `ntfsclone` efficiently clones (or copies) an NTFS filesystem to a file, copying only the used data. (Unused blocks are represented by zeros in a sparse file, so they don't take up space.)

You can't back up a Windows system using the live method. While you can easily download a Windows version of `tar`, and `tar` is available on the Knoppix CD, `tar` does not preserve Windows ACLs.

Step 3: Boot the System from Alternate Media

In order to easily recover your operating system, you need a limited root shell, which you get when you boot into Knoppix.

Once you boot into Knoppix, you can perform the rest of this procedure from a fully functional root shell.

Step 4: Restore the Boot Block Information

The boot block is a special part of the disk that loads the operating system by its “bootstraps,” hence the name. It contains just enough executable code to locate and load the kernel. On the Intel platform, this boot block is referred to as the master boot record, and you can restore it with `dd`, using the backup we created in the previous step.

> **Tip ✎**
>
> If you are using the alt-boot full image method, you can skip this step because the MBR is included in your image.

If you previously backed up the MBR and partition table using `dd`, as explained earlier, you can now use `dd` to restore the MBR and partition table by running the following command. Since the file */backups/mbr* contains the MBR, partition table, and MBR signature, restoring this file to the hard drive partitions it just like the one you backed up and makes it bootable. Once this is done, you’re ready to restore the operating system.

```
# dd if=/backups/mbr of=/dev/hda bs=512 count=1
```

In order to get Knoppix to recognize without a reboot that we had recovered the MBR, we found it necessary to actually run `fdisk /dev/hda` and then choose `w` to write the partition to disk. A reboot works as well but takes longer.

Restoring to Larger Hard Drives

The following procedure can be used to restore a smaller hard drive to a larger hard drive; you simply end up with a large unused section at the end of the drive. You can then extend or reorganize your partitions to use the extra space inside the Knoppix environment. QTParted is available in Knoppix, and it works a lot like the well-

known Partition Magic, automatically expanding and contracting partitions and filesystems as needed. At writing, QTParted does not support NTFS, so you need to expand any NTFS partitions manually. (You still need to do it inside Knoppix because you cannot expand a drive in Windows if it contains the swap file.) It's relatively simple, though.

This procedure works only if the partition you need to expand is the last (or only) partition on the hard drive. If there are any partitions after it, you will need to use a commercial solution such as Partition Magic (that is, until QTParted fully supports NTFS).

Use `fdisk` on the Knoppix CD to access the partition table on the new hard drive. Display the current partition table using the P key. Take note of the starting and ending cylinders of the partition you plan to extend, whether or not it is bootable, and the partition type. Also take note of the total number of cylinders in this hard drive and whether there are any partitions after the one you want to expand. If the partition you want to expand is *not* the last partition on the drive, you cannot use this procedure.

Delete the partition you plan to expand using the D key.

Use the N key to create a new partition starting with the same starting cylinder as the original cylinder and ending with the last cylinder on the drive, which you noted in Step 1.

If the partition in question was bootable, you'll need to make it bootable again. Do this by pressing the A key and entering the number of the partition in question.

You also need to set the partition type using the T key followed by the L key to list the partition types, and

locate the hex code for the partition type you saw in Step 1. Specify the partition type using its hex code.

If you're sure you did all those steps correctly, write the partition table using the W key and exit `fdisk`. (If you're not sure, you can quit `fdisk` at this point without doing any damage.)

Run the `ntfsresize` command against the partition (for example, `ntfsresize/dev/hda1`). If you confirm you want it to do so, it will automatically grow the NTFS partition to the end of the drive. You're done!

Step 5: Partition and Format the New Root Drive

The steps in the previous section restore both the MBR and the partition table, so there's no need to repartition the drive. All the partitions will already be created for you.

Tip ✎

If you're using the live or alt-boot filesystem methods, you can also restore from a smaller drive to a larger drive. Additionally, you can rearrange partitions as you see fit.

Linux, FreeBSD, and Solaris X86

If you're using either the live or alt-boot filesystem methods, you now need to create filesystems on the drive using the `mkfs` command. You need the backed up `fstab` file to know which filesystem types each partition is supposed to have.

If you're using the alt-boot full image or alt-boot partition image methods, you don't need to worry about formatting the drive; it happens automatically when you restore from the image.

Windows

If you're using the alt-boot full image or alt-boot partition image methods, you don't need to worry about formatting the drive; it happens automatically when you restore from the image. If you're using the alt-boot filesystem method, the drive is automatically formatted when you restore using `ntfsclone`.

Step 6: Restore the OS to the New Root Drive

First, you must have access to the backed-up data. Our examples use an NFS directory. You may have to do one of the following:

- Mount a ZIP or Jaz drive as a filesystem.
- Install a second hard drive of equal or greater size.
- Load network and NFS drivers, configure your network interface, and mount an NFS filesystem.
- Load network and SMB/CIFS drivers, configure your network interface, and mount an SMB/CIFS filesystem.

Our example backups were sent to an NFS filesystem. The drivers and tools that were needed to do this are all available on Knoppix.

Assumptions

The following sections provide procedures for each of the four methods we've discussed: alt-boot full image, alt-boot partition image, alt-boot filesystem, and live. The procedures are based on a few assumptions, which are explained here:

We assume DHCP is available

> We assume for our examples that your environment has a DHCP server supplying IP addresses so that you do not need to choose an IP address and configure the interface each time you boot into Knoppix. If DHCP is not available, you need to manually configure your IP address. To do this, run the following commands:
>
> ```
> $ su -
> # ifconfig eth0 ipaddress up
> # route add default gwipaddress-of-gateway
> ```
>
> You then need to use IP addresses to communicate with other machines, or enter a valid IP address of a DNS server into the */etc/resolv.conf* file (in the form of `nameserver` *`ipaddress-of-dns-server`*.)

We assume you're using NFS

> We assume for our examples that you have an NFS server in your environment with sufficient space to store the data on your server. If you'd like to back up your data to a Windows share instead, you can use this command:
>
> ```
> # mount -t smbfs -o
> username=administrator,password=PASSWORD//servername/share /backups
> ```

Alt-Boot Full Image Method

This example is the simplest of all the procedures, making it the most likely procedure for Windows users who are not familiar with Linux. This method requires the the system to be offline (booted into Knoppix), and it works regardless of what operating system you're using. It does require the system to be down and requires enough space for a copy of the entire root drive.

Create the Bare-Metal Backup

Use the following steps to create a bare-metal backup of your system.

Back up the important metadata

This method does not require this step because the metadata is backed up and restored when you back up the entire hard disk as one image.

Boot the system from alternate media

First, place the Knoppix CD into the drive and reboot, booting you into Knoppix. By default, Knoppix starts KDE (a windowing environment) as user *knoppix*. After switching to the root user (which has no password initially), create a mount point, and mount an NFS directory as `/backups`:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

Tip ✎

See the previous section “Assumptions” if either DHCP or NFS isn’t available.

Back up the operating system with a native utility

You can back up the entire disk using `dd` to a file in the NFS directory. This command specifies to back up the root hard drive (*/dev/hda*) to a file called */backups/hda.dd*.

```
# dd if=/dev/hda of=/backups/hda.dd
```

Alternatively, if you want to save space, you could run this command. Depending on where your bottleneck is, this may speed up or slow down the backup.

```
# dd if=/dev/hda |gzip –c > /backups/hda.dd.gz
```

Perform a Bare-Metal Recovery

Use the following steps to recover your system from bare metal.

Trash Your Hard Drive!

If you’re trying to test the recovery of your operating system hard drive, you might want to trash it before you start to make sure the procedure works. (We assume you’re working on a test system.) The first `dd` command blows away both the boot block and the disk’s partition information. The other commands mess up the first 50 MB of each partition. Although the root filesystem is still there, you couldn’t find it if you wanted to!

Boot into Knoppix, and then run the following commands:

```
# dd if=/dev/zero of=/dev/hda bs=512 count=1
# dd if=/dev/zero of=/dev/hda1 bs=1024k count=50
# dd if=/dev/zero of=/dev/hda2 bs=1024k count=50
# dd if=/dev/zero of=/dev/hda3 bs=1024k count=50
# dd if=/dev/zero of=/dev/hda4 bs=1024k count=50
# reboot
Boot Failure
Insert Boot diskette in A:
Press any key when ready.
```

Boot the system from alternate media

The first step in recovering this system is to place the Knoppix CD into the CD drive and boot the system. As before, open a terminal window, and switch to the root user, and then mount your NFS directory:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

Tip ✎

See the previous section “Assumptions” if either DHCP or NFS isn’t available.

Restore the boot block information

This step is accomplished when you perform the `dd` of the entire disk.

Prepare the new root drive

This step is accomplished when you perform the `dd` of the entire disk.

Restore the operating system

To restore the entire disk, run this command:

```
# dd if=/backups/hda.dd of=/dev/hda
```

If you used the `compress` command during backup, you should use this command to restore instead. Depending on where your bottleneck is, this may speed up or slow down the restore.

```
# gzip –dc /backups/hda.dd.gz|dd of=/dev/had
```

Tip ✎

All you have to do now is remove the Knoppix CD and reboot the system. Your system should be fully operational. Wasn’t that easy?

Alt-Boot Partition Image Method

This example uses dd to back up and recover a complete system at the partition level. It requires the system to be offline (booted into Knoppix), and it works regardless of your operating system. It's slightly more complex than the alt-boot full image method and still requires the system to be down during a backup, but it can be faster if your disk is partitioned correctly. You can save a lot of space and time if you partition your hard disk so that one partition contains the operating system and applications, and then apply this procedure only to that partition. (You would back up the other partitions with regular filesystem backup tools.)

Create the Bare-Metal Backup

Use the following steps to create a bare-metal backup of your system.

Back up the important metadata

Back up the MBR by running the following command:

```
# dd if=/dev/hda of=/backups/mbr bs=512 count =1
```

Boot the system from alternate media

Place the Knoppix CD into the drive and reboot into Knoppix. By default, Knoppix starts KDE (a windowing environment) as user *knoppix*. After switching to the root user (which has no password initially), create a mount point and mount an NFS directory as */backups*:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

> **Tip ✎**
>
> See the previous section “Assumptions” if either DHCP or NFS isn’t available.

Back up the operating system with a native utility

You then back up the operating system to the NFS directory using `dd`. You can find which partitions you need and back up just those partitions using these commands:

```
# fdisk -l

Disk /dev/hda: 41.1 GB, 41174138880 bytes
255 heads, 63 sectors/track, 5005 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1          13      104391   83  Linux
/dev/hda2              14        4874    39045982+  83  Linux
/dev/hda3            4875        5005     1052257+  82  Linux
swap / Solaris

# dd if=/dev/hda1 of=/backups/hda1.dd
#dd if=/dev/hda2 of=/backups/hda2.dd
```

> **Tip ✎**
>
> In our example, we did not back up */dev/hda3* because it is the swap partition and has no data.

Alternatively, you can also use compression. Depending on where your bottlenecks are, this may speed things up or slow them down.

```
# dd if=/dev/hda1| gzip –c > of=/backups/hda1.dd.gz
#dd if=/dev/hda2| gzip –c > of=/backups/hda2.dd.gz
```

You can place an & (ampersand) at the end to allow the backups to occur simultaneously by placing them in the background. However, this requires you to monitor those processes to ensure that they complete before you reboot.

Perform a Bare-Metal Recovery

Use the following steps to recover your system from bare metal.

Boot the system from alternate media

The first step in recovering this system is to place the Knoppix CD into the CD drive and boot up the system. A check of the partition table at this point shows the following:

```
# fdisk -l

Disk /dev/hda: 41.1 GB, 41174138880 bytes
16 heads, 63 sectors/track, 79780 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Disk /dev/hda doesn't contain a valid partition table
```

As before, open a terminal window and switch to the root user, then mount your NFS directory:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

> **Tip ✎**
>
> See the previous section “Assumptions” if either DHCP or NFS isn’t available.

Restore the boot block and prepare the new root drive

For restoring by partition, you need to restore the MBR and partition table and then restore each partition. You can restore the MBR and partition table by running the following command:

```
# dd if=/backups/mbr of=/dev/hda bs=512 count =1
```

In order to get Knoppix to recognize without a reboot that we had recovered the MBR, we found it was necessary to actually run `fdisk /dev/hda` and then choose `w` to write the partition to disk. A reboot works as well but takes longer.

Restore the operating system

You are now ready to actually restore the operating system. Use `dd` in the reverse order that you used to back up the operating system:

```
# dd if=/backups/hda1.dd of=/dev/hda1
#dd if=/backups/hda2.dd of=/dev/hda2
```

If you used the compression option in the backup, you should use these commands:

```
# gzip –dc /backups/hda1.dd.gz |dd of=/dev/hda1
#gzip –dc /backups/hda2.dd.gz |dd of=/dev/hda2
```

Again, we knew that there was nothing of value in */dev/hda3*.

You can place an & at the end to allow the restores to occur simultaneously by placing them in the background. However, this requires you to monitor those processes to ensure that they complete before you reboot.

All you have to do now is remove the Knoppix CD and reboot.

Live Method

This example uses `tar` to back up and recover the system at the partition level, which allows the system to be online. The procedure was tested on a Linux system. We did not test this on a Windows system because we could not identify a single utility available in Windows and Knoppix that preserves ACLs. Your experience may be different.

Create the Bare-Metal Backup

Use the following steps to create a bare-metal backup of your system.

Back up the important metadata

Back up the MBR by running the following command:

```
# dd if=/dev/hda of=/backups/mbr bs=512 count=1
```

Back up the operating system with a native utility

Create a mount point, and mount a NFS directory as */backups*:

```
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

> **Tip ✎**
>
> See the previous section “Assumptions” if either DHCP or NFS isn’t available.

You can use whatever method you like to back up the operating system at this point. For this example, we chose

`tar`. Our example has one partition mounted as */boot*, and the rest of the OS on */*.

```
# cd /boot
# tar cf /backups/boot.tar .
# cd /
#tar cf /backups/system.tar --exclude /mnt --exclude /proc --exclude /boot --
exclude /backups .
```

Alternatively, you could use compression. This may speed up or slow down the backup, depending on where the bottleneck is.

```
# cd /boot
# tar cfz /backups/boot.tar.gz .
# cd /
#tar cfz /backups/system.tar.gz --exclude /mnt --exclude /proc --exclude /boot --
exclude /backups .
```

Perform a Bare-Metal Recovery

Use the following steps to recover your system from bare metal.

Boot the system from alternate media

The first step in recovering this system is to place the Knoppix CD into the CD drive and boot up the system. A check of the partition table at this point shows the following:

```
# fdisk -l

Disk /dev/hda: 41.1 GB, 41174138880 bytes
16 heads, 63 sectors/track, 79780 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Disk /dev/hda doesn't contain a valid partition table
```

As before, open a terminal window and switch to the root user, then mount your NFS directory:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

> **Tip ✎**
>
> See the previous section “Assumptions” if either DHCP or NFS isn’t available.

Restore the boot block information and prepare the new root drive

For restoring by partition, you need to restore the MBR and partition table and then restore each partition. Restore the MBR and partition table by running the following command:

```
# dd if=/backups/mbr of=/dev/hda bs=512 count =1
```

In order to get Knoppix to recognize without a reboot that we had recovered the MBR, we found it was necessary to actually run `fdisk /dev/hda` and then choose `w` to write the partition to disk. A reboot works as well but takes longer.

You also need to prepare the partitions for a filesystem restore. If the *fstab* backup shows they were ext2 filesystems, run the following commands. (Obviously, this part varies based on the types of filesystems you’re using.)

```
# mkfs –t ext2 /dev/hda1 -L /boot
#mkfs –t ext2 /dev/hda2 -L /
```

Restore the operating system

You are now ready to restore the operating system. Mount the partitions:

```
# mount /dev/hda1
#mount /dev/hda2
```

> **Tip ✎**
>
> Knoppix, by default, creates a mount point for each partition it sees, so you should already have */mnt/hda1* and */mnt/hda2* available, and the command `mount /dev/hda1` mounts that partition to */mnt/hda1*. If you don't, you need to create them first.

`cd` to the location of the new root filesystem and run the restore commands.

```
# cd /mnt/hda1
# tar xpkf /backups/boot.tar
# cd /mnt/hda2
#tar xpkf /backups/system.tar
```

Or, if you chose compression, run these commands:

```
# cd /mnt/hda1
# tar xpkfz /backups/boot.tar.gz
# cd /mnt/hda2
#tar xpkfz /backups/system.tar.gz
```

All you need to do now is eject the Knoppix CD and reboot.

Alt-Boot Filesystem Method

This example uses `tar` to back up and recover Linux partitions and `ntsfclone` to back up and recover Windows partitions. This requires the system to be offline (booted into Knoppix). This process should work with Linux and any Windows version that uses NTFS.

Create the Bare-Metal Backup

Use the following steps to create a bare-metal backup of your system.

Back up the important metadata

Back up the MBR by running the following command:

```
# dd if=/dev/hda of=/backups/mbr bs=512 count =1
```

This procedure also requires you to know which partitions are which format, especially the Windows partitions. You should record this data by backing up the output of `fdisk -l`, backing up the *fstab* file, and making a text file that contains any additional necessary information.

In our example, */dev/hda1* is */boot*, */dev/hda2* is */*, and */dev/hda3* is the Windows partition.

Boot the system from alternate media

Insert the Knoppix CD, boot into Knoppix, and open up a terminal window. Knoppix, by default, starts up KDE (a windowing environment) as user *knoppix*. You then need to switch to the root user (which has no password initially).

```
knoppix@0[knoppix]$ su -
```

Back up the operating system with a native utility

The first thing you have to do for this procedure to work is to mount an NFS directory as `/backups`:

```
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

> **Tip ✎**
>
> See the previous section “Assumptions” if either DHCP or NFS isn’t available.

You then need to mount the various partitions as filesystems.

```
# mount /dev/hda1
#mount /dev/hda2
```

You can use whatever method you like to back up the operating system at this point. For this example, we chose `tar` for the Linux partitions and `ntfsclone` for the Windows partition. While you could use `tar` on the Windows partition as well, we felt that `ntfsclone` was more likely to preserve any Windows ACLs.

Our example has one partition mounted as */boot*, and the rest of the OS on */*:

```
# cd /mnt/hda1
# tar cf /backups/boot.tar .
# cd /mnt/hda2
#tar cf /backups/system.tar --exclude /mnt --exclude /boot --exclude /backups .
```

Alternatively, you could also use compression. This may speed up or slow down the backup, depending on where the bottleneck is.

```
# cd /mnt/hda1
# tar cfz /backups/boot.tar.gz .
# cd /mnt/hda2
```

```
# tar cfz /backups/system.tar.gz --exclude /mnt --exclude /proc --exclude/boot --
exclude /backups .
```

Now you need to back up the Windows partition. `ntfsclone` is not a filesystem utility per se because it does not back up on the file level. It's really an image backup utility that understands NTFS filesystems.

```
# ntfsclone --save-image --output /backups/ntfs-clone.img /dev/hda3
```

Perform a Bare-Metal Recovery

Use the following steps to recover your system from bare metal.

Boot the system from alternate media

The first step in recovering this system is to place the Knoppix CD into the CD drive and boot the system. A check of the partition table at this point shows the following:

```
# fdisk -l

Disk /dev/hda: 41.1 GB, 41174138880 bytes
16 heads, 63 sectors/track, 79780 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

Disk /dev/hda doesn't contain a valid partition table
```

As before, open a terminal window and switch to the root user, then mount your NFS directory:

```
knoppix@0[knoppix]$ su -
# mkdir /backups
#mount nfsserver:/data08/curtis /backups
```

Tip ✎

See the previous section "Assumptions" if either DHCP or NFS isn't available.

Restore the boot block and prepare the new root drive

For restoring by partition, you need to restore the MBR and partition table and then restore each partition. You can restore the MBR and partition table by running the following command:

```
# dd if=/backups/mbr of=/dev/hda bs=512 count =1
```

In order to get Knoppix to recognize without a reboot that we had recovered the MBR, we found it was necessary to actually run `fdisk /dev/hda` and then choose `w` to write the partition to disk. A reboot works as well but takes longer.

You also need to prepare the partitions for a filesystem restore. Since our *fstab* backup showed that */dev/hda1* and */dev/hda2* were ext2 filesystems, run the following commands:

```
# mkfs –t ext2 /dev/hda1 -L /boot
#mkfs –t ext2 /dev/hda2 -L /
```

You do not need to prepare the NTFS partition; it's restored with the `ntfsclone` command.

Restore the operating system

You are now ready to actually restore the operating system. We use `tar` to restore the Linux partitions and `ntfsclone` to restore the Windows partition. You need to mount the partitions:

```
# mount /dev/hda1
#mount /dev/hda2
```

Now `cd` to the location of the new root filesystem and run the restore commands:

```
# cd /mnt/hda1
# tar xpkf /backups/boot.tar
```

```
# cd /mnt/hda2
#tar xpkf /backups/system.tar
```

Or, if you chose compression, run these commands:

```
# cd /mnt/hda1
# tar xpkfz /backups/boot.tar.gz
# cd /mnt/hda2
#tar xpkfz /backups/system.tar.gz
```

Now you need to restore the Windows partition that's on */dev/hda3*. Use the `ntfsclone` utility to do that:

```
# ntfsclone --restore-image /backups/ntfs-clone.img  --overwrite /dev/hda3
```

Tip ✎

All you need to do now is eject the Knoppix CD and reboot.

Automate Bare-Metal Recovery with G4L

Now that we've discussed how to back up machines using a Linux boot disk, there is an easier solution that some users may find to their liking. G4L is a bootable CD-ROM that provides a menu-based system that automates making all the alternate-boot bare-metal backups we covered earlier. G4L is a freeware tool currently based on a 2.6 Linux kernel. It includes `dd`, `ntfsclone`, and `tar` but does not include `mount`, so backups to another host are done using FTP.

> **Warning ☠**
>
> G4L is an abbreviation for Ghost 4 Linux, but do not confuse it with another project by that name. G4L does not use Norton Ghost and uses the term *ghost* only in a generic sense; it was used long before Norton created its brand. The other project that you will find if you search the Internet for the phrase "Ghost4Linux" uses a network boot of Linux to run a simulated DOS environment that can run some versions of Norton Ghost. That is not how G4L works; it is its own complete utility.

Advantages of G4L

G4L is much easier to use than the method outlined in this chapter. For most users, no Linux knowledge is needed. Since FTP is a much more efficient way to transfer data than NFS or CIFS, it is typically 20 to 30 percent faster. Because G4L uses FTP, Windows-only environments can easily set up an IIS FTP server to store the backup images.

G4L can automatically compress the backup in stream, using light to heavy compression to improve backup speed. It also provides easy disk-to-disk duplication support. It supports SAN devices and can be used to do disk-to-disk backups. G4L also includes the `dd_rhelp` and `dd_rescue` utilities to recover a disk with bad sectors. G4L runs as a Microsoft Virtual PC Machine, so you can test it out in a safe environment.

Drawbacks of G4L

G4L does have some limitations. Since it is a Linux-based imaging solution, it may not have driver support for some hardware. It can only do backups over the network using FTP. It does not currently support NFS or SMB/CIFS mounts. Even though it is simple and fairly intuitive, its documentation is currently not as good as it could be. Hopefully, that will improve.

Setting Up G4L

Setting up G4L is very easy. Since you are backing up using FTP, you need an FTP server. This can be done easily using IIS, `war-ftpd`, or any other FTP server that you like. You need to create a username and password, and give that user permission to upload. You also need to create a directory large enough to hold the backups. You should test the FTP server by connecting from a desktop and uploading a file to the upload directory.

Next, you need to download the G4L ISO image and burn a CD of it. The G4L project page is http://sourceforge.net/projects/g4l. Once you have downloaded the image, you can burn it using a CD writing software package.

With the bootable G4L CD, you are ready to boot and back up a server. Insert the CD in a system and boot from it. The CD boots to a license and informational screen; once you press Enter to agree, you boot to a command line.

Using G4L

To start G4L, simply run the `g4l` command at the prompt:

```
bash-3.00# g4l
```

This displays the main menu. Here, you can choose Raw Mode or File Mode. The Raw Mode option is the one to use for most backups. File Mode requires you to set up a special server running `partimaged`. Don't worry, though; you can still do filesystem backups using `ntfsclone` in Raw Mode.

Raw Mode offers three choices: Network Use, Local Use, and Click' n' Clone. Choose the Network Use option. The Local Use option allows you to perform a raw backup or restore of a drive, or partition to a file on another drive. The Click' n' Clone option allows you to clone one disk to another.

Selecting the Network Use option displays a menu with 16 options. Don't panic; in most cases, you'll need only 3 to run a backup.

G4L defaults to using eth0 and DHCP to set network settings. These defaults are suitable for most environments, but if you need to change them, you can. The first option, `A: Pick Device`, allows us to select a different Ethernet adapter. `Option B: Config Device` allows you to set an IP address for that Ethernet device.

You must set options `D: Config FTP`, `E: Config useridpass`, and `F: Config Filename` before performing a backup or restore. `D: Config FTP` is the setting for the FTP server that

you are backing up to. `E: Config useridpass` is the user ID and password for the FTP server. (You enter *`username:password`*.) `F: Config Filename` specifies the name for the backup.

Additionally, if the directory on the FTP server you are backing up to is not */img*, set `P: Path to Image Directory`. You can also set the compression algorithm. `G: Toggle Compression` allows you to select `None`, `GZip`, `Lzop`, or `BZip2` compression. Without getting into a battle royal over which compression method is best, `lzop` produces the fastest backups. Most users should use `lzop` because the backup will be smaller and run faster on most modern Pentium I or later machines.

With those settings, you can now choose how and what to back up. `H: Backup` allows you to select a drive or partition to back up. Once you select the partition or drive and click OK, the backup runs. Similarly, selecting `I: Restore` restores the raw backup from FTP. If you want to perform a filesystem backup, select `N: NTFSCLONE Backup` and then select from the available NTFS partitions. `O: NTFSCLONE Restore` allows you to restore an NTFS partition backup.

After making or restoring a backup you can see the summary performance data by selecting the `T: Display Time` option.

Customizing G4L

It is also possible to customize G4L for your environment. The easiest way to do this is to use a Linux development environment. You can set the menus to have different defaults by mounting the ISO image as a loopback device and editing the config files before you burn the CD. You can set the defaults in the G4L file itself by changing the variables `netzip`, `server`, `useridpass`, `netimagename`,

`device`, and `ftppath`. The variables reflect in order the compression used, the FTP server IP address, the username and password for the FTP server, the default backup filename, the Ethernet device, and the FTP upload directory. You can also perform other customizations to the menus to remove choices that are not relevant to your environment.

Commercial Solutions

Aside from the free methods described in this chapter, there are a number of inexpensive commercial bare-metal recovery tools. Many of the enterprise-level backup products also provide bare-metal recovery capabilities. Here are some inexpensive (starting at $99) bare-metal recovery tools for Windows:

Acronis (http://www.acronis.com)
Ghost (http://www.symantec.com)
Paragon (www.drive-backup.com)
Winternals (http://www.winternals.com)

Ghost keeps snapshots of critical system files, allowing you to roll back to a known good configuration as long as your hard disk is functional. We found only one inexpensive tool for bare-metal recovery of Linux systems: Storix (http://www.storix.com). For a product that starts at $99, the number of recovery options it offers is incredible.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 12. HP-UX Bare-Metal Recovery

This chapter explains procedures that a system administrator would use to recover an HP-UX operating system in case of a complete system failure—when there is nothing left but bare metal. Hewlett Packard Ignite-UX provides a unique approach to system recovery by integrating bare-metal system recovery capabilities with general deployment tool sets within a client/server framework. The Ignite-UX `make_net_recovery` and `make_tape_recovery` tools may be used to create bootable system recovery archives that are intended to return the installed OS to new or original hardware in a disaster. The recovery archives may be stored across the network on disk or on local tape drives. Ignite-UX may be installed from HP-UX application CDs or downloaded free from http://software.hp.com.

Tip ✎

This chapter was contributed by Eric Stahl and Ron Goodwyn. Eric has 25 years in IT including stints with Hewlett Packard, the Space Shuttle Launch Control Center, and flight-testing the first F15E and B2 aircraft. Ron received his B.S. in computer science from Tuskegee University and has written a few articles on Ignite-UX system recovery in the past.

System Recovery with Ignite-UX

The original Ignite-UX recovery tool was called `make_recovery`. It provided system administrators with an efficient means for creating bootable backup tapes of the root volume group (containing the boot disk). For current

releases of Ignite-UX, `make_tape_recovery` obsoletes `make_recovery`, offering tighter integration with the Ignite-UX architecture and enhanced capabilities. `make_net_recovery` is also available, due to the increased appearance of compact rack mountable servers, increases in network speeds, and the availability of inexpensive disks.

> **Warning ☠**
>
> Starting with HP-UX 11.0, the `copyutil` tool previously provided on the Support Media and used to create HP-UX 9.x and 10.x system recovery images on tape should not be used beyond its current capacity as an offline diagnostics tool. Ignite-UX now provides enhanced support for the installation and recovery of customized HP-UX systems. `copyutil` is now part of the Offline Diagnostic Environment (ODE), an offline support tools platform for troubleshooting a system that is running without an OS or that cannot be tested with the online tools.

The `make_net_recovery` and `make_tape_recovery` commands must be run by the root user from the command line or accessed through a graphical or terminal user interface (GUI/TUI) on the Ignite-UX network server. You can use these commands on a running system, though it’s inadvisable to do so during periods of heavy activity when files, directories, and system configurations may be changing a lot.

Tip ✎

Ignite-UX supports both LVM (Logical Volume Manager) and VxVM (Veritas Volume Manager) as well as whole disk configurations.

Ignite-UX Overview

The installation and configuration of an Ignite-UX server is a prerequisite for network-based `make_net_recovery` and `make_tape_recovery` operations. Ignite-UX technology also allows system administrators to define new system configurations and then deploy them from a centralized server to clients that may be running different versions of HP-UX.

Before starting such operations, the Ignite-UX server must first be configured for network operations, and custom deployment configurations including the OS and applications must already be set up. Sometimes a build-and-fix approach is used with test systems to ensure that things work properly before deployment to a production environment. But once installation is under way, all systems can be installed in a consistent fashion within a few hours.

The design of the Ignite-UX toolsets is very modular and includes commands that can be run independently of one another. Many of the underlying Ignite-UX tools are Unix shell scripts—advantageous in a Unix environment because it makes management tasks much easier for the system administrator. The primary tools that run beneath `make_net_recovery` and `make_tape_recovery` are `save_config` and `make_sys_image`. `make_tape_recovery` uses the `make_medialif` command (on HP9000 servers) or `make_ipf_tape` (on HP Integrity servers) to create the boot

components that are eventually written to tape. Most of these commands have manpages with useful examples. Reading the manpages and examining the contents of the shell scripts are a good way to learn underlying functionality.

The `make_sys_image` command may be viewed as the workhorse of the system recovery tools; it actually creates the system recovery archive. The `save_config` command captures information about the personality of the system. Conceptually, `make_sys_image` can be viewed as providing the bits while the `save_config` command can be viewed as the tool providing the bucket.

You may also use Ignite-UX and HP-UX tools to construct customized bootable install media in CD and DVD format. For DVDs, ISO installation images are created with the `mkisofs` command and then written to DVD with the `growisofs` command. CD installation images start in either HFS or CDFS filesystem format, and are finally written to CD with the `cdrecord` command. The `make_medialif` command helps build bootable CD or DVD media for both HP9000 and HP Integrity systems.

The system recovery model for HP-UX evolved from the general Ignite-UX network deployment model. Instead of installing generic operating system images, the `make_net_recovery` and `make_tape_recovery` commands create images based on the contents of the filesystems. Some filesystems are deemed as essential by the Ignite-UX recovery tools to constitute a functional system. The default essentials file list is located at */opt/ignite/recovery/mnr_essentials*. A user-defined version of the essentials file is used if it exists and is located at */var/opt/ignite/recovery/mnr_essentials*. The *mnr_essentials* file is not a list of all the files that will be archived (that file, *flist*, is discussed later). It lists only the minimally required essential files.

Warning ☠

Modifications to *mnr_essentials* should be avoided unless the additions are truly essential to a system recovery.

The recovery archive contains machine-specific configuration information such as hostname, IP address, networking information, and password files. Ignite-UX recovery mode enforces rules, ensuring that these files are restored as they exist in the system recovery archive. The configuration information may be modified before or during recovery operations. Recovery archives are written to a network-based Ignite-UX server by `make_net_recovery` or to local tape by `make_tape_recovery`.

Tip ✎

It is very important to note that neither the `make_tape_recovery` or `make_net_recovery` commands take the place of standard tape backup practices. The system recovery tools provided by Ignite-UX for regaining a bootable OS should be part of a total backup and disaster recovery strategy.

Network Services and Remote Boot Protocols

Network configuration and troubleshooting is perhaps the most involved aspect of Ignite-UX system recovery operations. Comprehensive instructions on Ignite-UX configuration and usage may be found online at http://docs.hp.com. Search the keyword `ignite`, and locate the most recent *Ignite-UX Administration Guide* in PDF or HTML

format. Remember that `make_tape_recovery` may be used as soon as Ignite-UX is installed on a system without having to configure an Ignite-UX server.

Ignite-UX services and protocols such as BOOTP, the Remote Maintenance Protocol (RMP), TFTP, and NFS have been available on Unix for many years. When a network recovery archive is created, the archive is written via the network in compressed format to the Ignite-UX server across NFS mount points. Optional archive transfer methods include `remsh` and `ftp` (refer to the `instl_adm(4)` manpage). Client hardware is uniquely recognized by the Ignite-UX server using the hardware (MAC) link-level addresses of network interfaces. During the recovery operations of a network archive, and depending upon system and network architecture, a remote client boot may utilize BOOTP, PXE (the Intel Preboot Execution Environment), RMP, TFTP, and NFS.

HP Integrity client systems running PXE communicate with network boot servers only through ports 67 and 68. HP9000 client systems using BOOTP or RMP may communicate with network boot servers either through ports 67 and 68 or through ports 1067 and 1068.

Ignite-UX supports RMP remote booting for older HP 700 series workstations (prior to the 712 series) that do not use BOOTP. `rbootd` is the remote boot server daemon that is part of the core HP-UX OS. Running on the Ignite-UX server, `rbootd` acts as a forwarding agent, formatting RMP boot requests into BOOTP boot requests and sending them to the server's `bootpd` or `instl_bootd` daemons. `rbootd` also formats information from BOOTP to RMP for communication back to the client.

BOOTP is the remote boot protocol. `bootpd` is the HP-UX boot protocol server daemon and is part of the core HP-UX OS. It is both a DHCP and BOOTP protocol server, using

ports 67 and 68 for communication between servers and either HP9000 or HP Integrity clients. It is configured using the */etc/dhcptab* file for DHCP configuration options (the dynamic allocation of IP addresses and hostnames), and the */etc/bootptab* file for BOOTP configuration options (containing fixed entries for specific clients on the network).

Prior to HP-UX 11.23, `bootpd` was configured through the */etc/bootptab* file for an Ignite-UX server to accept the remote boot requests of specific HP Integrity clients. Beginning with HP-UX 11.23, new options became available for `bootpd` through the */ etc/dhcptab* file so that the IP addresses in DHCP device pool groups are now available to anonymous HP Integrity clients booting on the network. This is advantageous if the Ignite-UX server is being otherwise used as both a DHCP and BOOTP network server.

`instl_bootd` is the boot protocol server daemon included with Ignite-UX that may be used in place of `bootpd`. It is configured through the */etc/opt/ignite/instl_bootptab* file. `instl_bootd` initially used only HP-specific ports 1067 and 1068 for communication between HP9000 servers and clients. Beginning with Ignite-UX 4.2, the `instl_bootd` boot protocol server daemon became configurable for supporting ports 67 and 68 and therefore HP Integrity `PXE` clients in addition to HP9000 BOOTP clients.

You may enable one or both of `bootpd` and `instl_bootd` on your Ignite-UX network server, and `instl_bootd` may be configured in one of two modes. This means that there are four possible network boot server configurations for use with Ignite-UX. Each configuration is implemented through changes made to the */etc/inetd.conf* file as shown later in this chapter.

The primary considerations for choosing a particular network boot server configuration are the versions of HP-UX and Ignite-UX being run, whether there are HP9000 and/or HP Integrity clients on the network, whether DHCP (through `bootpd`) is used with Ignite-UX, or whether DHCP is already being used on the server and should not be disabled.

You may enable `bootpd` for communication with either HP9000 or HP Integrity clients by uncommenting the following line in */etc/inetd.conf*. There should be no other lines uncommented for either `bootpd` or `instl_bootd`.

```
bootps dgram udp wait root /usr/lbin/bootpd bootpd
```

To enable `instl_bootd` to communicate with HP9000 clients only, comment out the `bootps` line from the */etc/inetd.conf* file, and add the following `instl_bootd` line:

```
#bootps dgram udp wait root /usr/lbin/bootpd bootpd
instl_boots dgram udp wait root /opt/ignite/lbin/instl_bootd
instl_bootd
```

To enable `instl_bootd` to communicate with either HP9000 or HP Integrity clients, add the following `instl_bootd` line to */etc/inetd.conf*:

```
#bootps dgram udp wait root /usr/lbin/bootpd bootpd
#instl_boots dgram udp wait root /opt/ignite/lbin/instl_bootd
instl_bootd
bootps dgram udp wait root /opt/ignite/lbin/instl_bootd
instl_bootd
```

For this configuration, it must be verified that the `bootpd` and `instl_bootd` lines shown in the previous two examples are commented out.

To configure both boot protocol server daemons so that `instl_bootd` monitors for HP9000 clients, and `bootpd` monitors for HP9000 or HP Integrity clients, use the following two uncommented lines in */etc/inetd.conf*.

```
bootps dgram udp wait root /usr/lbin/bootpd bootpd
instl_boots dgram udp wait root /opt/ignite/lbin/instl_bootd
instl_bootd
#bootps dgram udp wait root /opt/ignite/lbin/instl_bootd
instl_bootd
```

Before forcing `inetd` to read this new *inetd.conf* configuration, use the `netstat` command to verify that there are no running instances of `bootpd` on ports 67 or 68:

```
# netstat -an | grep -e "\.67" -e "\.68"
```

Issue this command to tell `inetd` to reread */etc/inetd.conf* and apply any changes:

```
# inetd -c
```

Differences Between HP Integrity and HP9000 Clients

After initiating a remote boot from either a HP9000 or HP Integrity client across the network from an Ignite-UX server, an Ignite-UX TUI is displayed on the client console. You may select the needed archive with the description specified during archive creation using the TUI. After moving through the various TUI tabs, press the Go button to continue. You may then monitor the status of the client's archive recovery from the Ignite-UX server's interface.

Prior to the appearance of the Ignite-UX TUI on the client console, the remote boot functionality of HP Integrity systems was different from that of HP9000 systems. And those differences must be taken into account when configuring and implementing an Ignite-UX environment.

From the system administrator's perspective, there are two primary differences:

- The initial choices in Ignite-UX server configuration to accommodate a mixed HP Integrity and HP9000 client

environment. These were discussed in the previous section.

- The console commands and screen presentations used on remotely booting HP Integrity and HP9000 clients.

When booting remotely, HP9000 systems (including the older 700 series workstations) may request services from specific Ignite-UX BOOTP servers using firmware-level Boot Console Handler (BCH) commands. We accessed the `BCH` prompt in the following command examples by interrupting the startup process of a HP9000 system. From there you can issue boot commands across the network to an Ignite-UX boot server.

The `install` keyword has special meaning in HP9000 BCH commands. It specifies that you want to use a BOOTP boot protocol server daemon. By excluding the `install` keyword, the following HP9000 BCH command searches the network for boot servers with `install_bootd` daemons:

```
BCH> search lan
```

Using the `install` keyword, the next command issues a remote boot request to the IP address of an Ignite-UX server that is known by the system administrator to hold the needed recovery archive and is enabled for network BOOTP service with `bootpd`.

```
BCH> boot lan.192.168.10.10 install
```

The next HP9000 BCH command example searches the network for available boot servers (including Ignite-UX boot helpers), using the `install` keyword to specify `bootpd` configured servers. A list of responding boot servers is presented, with each one referenced by a selectable `Path Number`.

```
BCH> search lan install
```

```
Searching for potential boot device.
This may take several minutes.

To discontinue, press ESCAPE

Path Number     Device Path                 Device      Type

P0              LAN.10.128.70.127.3.254     courage
P1              LAN.10.128.70.128.3.254     ibanez

BCH>boot P1
```

HP Integrity systems using PXE have no equivalent commands for specifying the Ignite-UX network boot server. Instead, Ignite-UX boot servers are configured in specific ways to handle the requests of both HP9000 clients, individual HP Integrity clients, and/or the requests of multiple and anonymous HP Integrity clients.

In this example, a system administrator used `telnet` to log in to the Management Processor (MP) of an HP Integrity rx7620 system that may or may not have a running or bootable OS. For a variety of both HP9000 and HP Integrity systems, this type of remote console access is a convenient way to avoid a long drive into a data center.

```
# telnet eiger-mp1
Trying...
Connected to eiger-mp1.rose.hp.com.
Escape character is '^]'.
Local flow control off

MP login: Admin
MP password:


                        Welcome to the

                    rx7620 Management Processor

(c) Copyright 1995-2004 Hewlett-Packard Co., All Rights Reserved.

                        Version A.7.002
```

```
    MP MAIN MENU:

         CO: Consoles
        VFP: Virtual Front Panel (partition status)
         CM: Command Menu
         CL: Console Logs
         SL: Show Event Logs
         HE: Help
          X: Exit Connection

[mp] MP>
```

Using this menu, you can move between multiple bootable hardware partitions (nPars) configured on a server, reset systems, or bring up the console of an already booted OS.

In this example, the bootable hardware partition (nPar) being targeted for OS recovery operations has been reset and the startup process interrupted to bring up the Intel Extensible Firmware Interface (EFI) on the console. In some ways similar to BCH on HP9000, EFI provides firmware-level access to boot utilities on HP Integrity systems. It allows any EFI OS loader to be selected from any supported boot medium. Be sure to use the proper ANSII terminal emulation with the EFI menu interface or an alternative such as vt102.

After a system reset, and if the `autoboot` flag is not set, devices are mapped, and the EFI Boot Manager menu appears. If the `autoboot` flag is set, the console displays the message `Press Any Key to interrupt Autoboot`. From there, maneuver around the EFI Boot Manager menu using either the up and down arrow keys, or the ^ (caret) and V keys.

```
EFI Boot Manager ver 1.10 [14.61]  Please select a boot option

    HP-UX Primary Boot: 0/0/0/3/0.6.0
    HP-UX HA Alternate Boot: 0/0/0/3/0.5.0
    Acpi(HWP0002,8C)/Pci(1|0)/Pci(4|0)/Mac(00306EF397E9)
    EFI Shell [Built-in]
```

```
    Boot Option Maintenance Menu


    Use ^ and v to change option(s). Use Enter to select an
option
```

In this example, there is an existing option for booting through a particular network interface, identified here through Link Level MAC address 00306EF397E9. To add, remove, or otherwise modify a boot option, select Boot Option Maintenance Menu, followed by an option such as “Add a Boot Option.” Finally, save the changes to NVRAM.

Planning for Ignite-UX Archive Storage and Recovery

An Ignite-UX client system contains a subset of the software installed on the Ignite-UX server. The primary difference between the client and the server is that the server will be used to manage, among other things, the recovery operations of one or more remote clients. This primarily involves the enabling of additional network services, consideration of additional disk space to store recovery archives, and planning for the remote boot procedure that is used during recovery.

Considerations for the Remote Booting of Clients

Here are the initial considerations for remotely booting an HP-UX client system for Ignite-UX recovery:

Is there a running OS on the client, and does it have networking capabilities?

If there is a running OS with networking capabilities, the network boot of the client may be initiated from the Ignite-UX server with a recovery image being pushed across the network to the client. In this scenario, the Ignite-UX server uses the `bootsys` command to initiate the OS recovery to one or more clients, without having to interact with the console of each client. `bootsys` copies the Ignite-UX install kernel and root filesystem to the client, configures the client to boot automatically from this kernel, and reboots. The archive recovery is then performed across the network.

If the client has no running OS communicating with the network, it must be booted from either a directly connected

or a remote console to pull a recovery image across the network from the Ignite-UX server. When booting an HP Integrity client across the network in this manner (pull operation), any EFI-supported network interface may be used. When booting an HP9000 client in a pull operation, the client's built-in network interface must be used; otherwise, the HP9000 client needs to be recovered from a local recovery image such as a `make_tape_recovery` tape, custom CD/DVD Ignite-UX bootable recovery image, or other OS installation media. The latter options, of course, assume that there is a tape or CD/DVD drive available for the client systems.

Are the client and Ignite-UX servers on the same network subnet?

If the client and Ignite-UX boot server are on different subnets, either an Ignite-UX boot helper must be configured on the same subnet as the client, or as discussed earlier, local tape or CD/DVD operations should be used for the initial boot prior to connecting with the Ignite-UX server.

The purpose of the boot helper is to provide the temporary IP address, install kernel, and install root filesystem. In other words, the initial network boot of a client to a boot protocol server must be across the same subnet. From that point, the client will be able to communicate with an Ignite-UX server on another subnet to download and install the recovery archive. Configuration of a boot helper system is detailed in the *Ignite-UX Administration Guide* at docs.hp.com.

How can I plan ahead for remote boot scenarios?

For client systems that might potentially fail in such a way as to require booting across the network from firmware using the client's local console and a specific network interface, it may be helpful to prepare ahead of time by

collecting the network information that will be needed during recovery.

In the following example, an HP Integrity system has five network interfaces, four of which are on a four-port network card. The primary interface used for the remote network boot is found at the hardware address `0/0/8/1/0/4/0`. The `rx` designation in the server model indicates that this is an HP Integrity system. The `lanscan` command output includes the hardware addresses and Link Level MAC addresses. The `ioscan` command output includes the HP part numbers for the interfaces.

```
# model
ia64 hp server rx7620
# lanscan
Hardware Station        Crd Hdw   Net-Interface  NM  MAC
HP-DLPI DLPI
Path     Address        In# State NamePPA        ID  Type
Support Mjr#
0/0/8/1/0/4/0 0x00306EF397E9  0   UP    lan0 snap0    1   ETHER
Yes     119
0/0/14/1/0/4/0 0x000E7F4FB144 1   UP    lan1 snap1    2   ETHER
Yes     119
0/0/14/1/0/5/0 0x000E7F4FB145 2   UP    lan2 snap2    3   ETHER
Yes     119
0/0/14/1/0/6/0 0x000E7F4FB146 3   UP    lan3 snap3    4   ETHER
Yes     119
0/0/14/1/0/7/0 0x000E7F4FB147 4   UP    lan4 snap4    5   ETHER
Yes     119
# ioscan -fnC lan
Class I H/W Path     Driver S/W State H/W Type Description
================================================================
==========
lan 0 0/0/8/1/0/4/0  igelan CLAIMED INTERFACE HP A6794-60001 PCI
1000Base-T
lan 1 0/0/14/1/0/4/0 btlan  CLAIMED INTERFACE HP A5506B PCI
10/100Base-TX 4 Port
lan 2 0/0/14/1/0/5/0 btlan  CLAIMED INTERFACE HP A5506B PCI
10/100Base-TX 4 Port
lan 3 0/0/14/1/0/6/0 btlan  CLAIMED INTERFACE HP A5506B PCI
10/100Base-TX 4 Port
lan 4 0/0/14/1/0/7/0 btlan  CLAIMED INTERFACE HP A5506B PCI
10/100Base-TX 4 Port
```

Sizing the Recovery Archive

It’s important to properly size the recovery archive. If the archive is large and you use `make_tape_recovery`, the archive could potentially span multiple tapes. This represents a disadvantage during both archive creation and recovery by requiring user interaction in a process that might otherwise occur unattended (for example, through a `cron` job). Many system administrators schedule tape creation during times when the system has reduced utilization, such as during the night and on weekends. For `make_net_recovery` archives, the added complexities would include calculations for required disk space as well as for offline storage space and tape backup completion times. Note that the number of successive archives to be retained on disk is specified either through the Ignite-UX server interface or from the command line using the `-n` option. The Ignite-UX parameter that records this information is called `save_num_archives` and is stored in the */var/opt/ignite/clients/client_name/recovery/defaults* file.

Another consideration involving the sizing of recovery archives involves the placement of filesystems into other volume groups that are normally found in the root volume group. Avoid the temptation to fragment required system directories and files across volume groups. The best practice when deciding what to include or not to include within the root volume group and within a recovery archive is to maintain a bootable system of a reasonable size completely within the root volume group.

Tip ✎

When creating an archive for the Ignite-UX server itself, consider excluding the collection of on-disk client recovery archives that are found by default under the */var* filesystem. Another example of a filesystem that might be excluded from a recovery archive is */home*. Consider the size of that filesystem in relation to the rest of the archive, how often the recovery archives are created, and if it would be acceptable following an OS recovery operation to restore the most recently saved */home* filesystem from normal tape backups.

`make_net_recovery` uses the network to store and retrieve archives via two NFS mount points. One mount point stores configuration information about the client; the other stores the archive itself. The archive does not need to be stored on the Ignite-UX server. The archive may be stored on any properly configured NFS server. In this configuration, the Ignite-UX recovery management process involves three different systems including the Ignite-UX server, the client, and (optionally) a remote archive server.

The default storage location for individual recovery archives is defined during the initial client configuration through the Ignite-UX server interface. You may choose alternate locations using the `–a` option to the `make_net_recovery` command. The Ignite-UX parameter that records this information is called `recovery_location`, and is stored in the */var/opt/ignite/clients/client_name/recovery/defaults* file. The directories and NFS export rules of an alternate location must be manually created before the alternate location may be used.

Configuring an Ignite-UX Network Server

Following the installation of Ignite-UX using the `swinstall` command, you begin the process of configuring an Ignite-UX network server the first time root brings up the server interface with the `/opt/ignite/bin/ignite` command.

> **Tip ✎**
>
> While the configuration of Ignite-UX is required before running the network-based `make_net_recovery`, `make_tape_recovery` may be used to create recovery archives to a local tape drive as soon as Ignite-UX is installed on an HP-UX system. The advantage to initiating `make_net_recovery` and `make_tape_recovery` operations from the Ignite-UX server is that the Ignite-UX software required for recovery archive operations is automatically installed onto the clients.

The initial “Welcome To Ignite-UX” screen has a Server Setup button and a “Tutorial and Demo” button, in case you would like an onscreen reference.

The Server Setup wizard steps the system administrator through the following activities:

1. Setting up temporary IP addresses that help boot client systems to an install kernel from the server in preparation for installation/recovery. It is important to verify that these temporary IP addresses are not already in use. One address is required for each anticipated simultaneous installation/recovery operation.

2. Setting up DHCP addresses (optional) used by the client systems after the initial remote boot from the server. The DHCP service is used only for client configurations that do not have predefined system hostnames and IP addresses.
3. Setting up a software depot containing core OS software that is used in the Ignite-UX deployment model. (This is not used in system recovery.)

The next time that the “Welcome To Ignite-UX” screen appears following configuration, select the OK button to display the main Ignite-UX server interface. From the main interface in the Options drop-down menu, select Server Configuration to review and modify a number of Ignite-UX server and configuration parameters. The Session Options control the behavior of the client systems during installation/recovery.

Select Actions→“Add New Client for Recovery”. After entering a client hostname, Ignite-UX uses either `remsh` or `ssh` to gather needed information from the client. Recovery software needed on the client is automatically installed or updated when working from the Ignite-UX server interface. If `make_net_recovery` is run from the command line of the client, and the recovery software is a different version than is on the server, an error may be generated. To manually initiate the update (not the installation) of recovery software on the client, refer to the `make_net_recovery` manpage for an example script that uses the `/opt/ignite/lbin/check_version` and SD-UX `swinstall` commands.

The previously mentioned new functionality for HP-UX 11.23 `bootpd` support of DHCP device pool groups requires the manual editing of the */etc/dhcptab* file using the new `dhcp_device_group` configuration options `re` and `ncid`. `re`

instructs the DHCP server to perform regular expression matching on the class-id. `ncid` indicates that the response from the server will not contain a class-id because BOOTP does not support the full PXE protocol.

Following is an example `dhcp_device_group` entry with the two new options that would be referenced by `bootpd` starting at HP-UX 11.23. The `bf` (boot file) option specifies that the network bootstrap program for EFI-partitioned boot disks be downloaded to continue booting (EFI is the Intel Extensible Firmware Interface used by HP Integrity systems). The other `dhcp_device_group` options in this example direct the IP address assignment.

```
dhcp_device_group:\
   re:\
   ncid:\
   class-id="PXEClient:Arch:00002:.*":\
   lease-time=300:\
   subnet-mask=255.255.255.0:\
   addr-pool-start-address=192.168.1.10:\
   addr-pool-last-address=192.168.1.20:\
   bf=/opt/ignite/boot/nbp.efi
```

For complete configuration details, refer to the most recent *Ignite-UX Administration Guide*, found online at http://docs.hp.com, as well as the manpages for the various Ignite-UX and HP-UX commands.

Recovery Archive Management

You may initiate the creation or restoration of system recovery archives from the Ignite-UX GUI or via the command line. If a graphics display is not available on the Ignite-UX server, use the TUI mode to run the `ignite` interface.

The `make_net_recovery` command may be initiated from the command line of the client, or through the Ignite-UX server interface. The `make_tape_recovery` command may

be initiated from either the command line on the client or through the Ignite-UX interface on the server. When initiated from the Ignite-UX server, `make_tape_recovery` executes on the client and writes to the client's local tape drive. When `make_tape_recovery` is initiated from the client's command line instead of from the Ignite-UX server interface, no NFS mounting is required nor is any Ignite-UX server configuration required. If the `make_tape_recovery -s` option is used from the client to specify an Ignite-UX server, then NFS mounting occurs, and it uses the configuration information stored from that client's last server-initiated `make_tape_recovery` session.

Following the initial Ignite-UX server configuration, you can continue to use the GUI or TUI server interface to manage network-based archives. Here are some of its advantages:

- There is centralized management and control of network-wide activities.
- Configuration settings made in the GUI or TUI are saved on the Ignite-UX server and are automatically loaded in subsequent runs.
- The required recovery tools are automatically installed and updated to the client systems.
- An icon representing the system being archived is graphically displayed; a status monitor that indicates progress of the archive creation from start to finish can also be displayed.
- Default directories are automatically created with related NFS mounts and `exportfs` configuration information that facilitates client/server interaction. If the directories used for storing archives are not the default (including being on a server different from the Ignite-UX server), this step is required to be performed

manually on both the Ignite-UX server and on the archive server.

Here are three scenarios in which users might consider using the command-line interface.

- The system recovery commands have been run once from the GUI, and the stored settings from that session must be overridden.
- The system recovery commands have been run once from the GUI, and future invocations of the commands should use those exact same settings as stored.
- The system recovery commands are being run for the first time, and as an experienced user, you want to see the output directly from the commands.

Implementation Example

The system administrator noticed that system logs are showing indications of hardware errors and filesystem corruption in the root volume group. Further investigation included the use of system hardware diagnostics to isolate the failure to the boot disk hardware. Now it’s time for the bare-metal recovery plan to be enforced to get the system back into fully operational use. The following example illustrates the use of `make_net_recovery`. (The `make_tape_recovery` steps would be similar.)

Before the point of failure, the system administrator created a recovery archive by taking these steps:

1. The administrator installed the Ignite-UX software onto a network server from the HP web site at http://software.hp.com or from the HP-UX Applications media.
2. Following installation, the administrator ran the `/opt/ignite/bin/ignite` command to launch the GUI/TUI interface and configure the Ignite-UX server.
3. Still in the Ignite-UX server interface, the system administrator entered the name of the client machine being archived. Once Ignite-UX connected to the client, it collected needed information and displayed an icon for that client.
4. The administrator right-clicked on the client icon and selected Create Network Recovery Archive. Ignite-UX displayed a sequence of screens with various instructions and tips.
5. Ignite-UX displayed the recovery archive defaults, and the administrator changed a few options, such as the maximum number of retained archives and an archive description. The system administrator verified that the destination directory had sufficient filesystem space to store the archive. The default destination directory name on the server was *var/opt/ignite/recovery/archives/<client_name>*.

Tip ✎

By default, Ignite-UX archives only the root volume group. Optionally, other disks and volume groups may be included in an archive, but it is a good practice to archive only the root volume group and to ensure that it contains only what is necessary to boot and run the OS.

1. After selecting the option to `Finish`, the archive creation began.
2. The icon color reflected success, errors, or warnings in the archive process. (Green indicates success while red and yellow indicate errors and warnings, respectively. To monitor progress of the archive creation in real time, right-click the client icon and select Client Status.)
3. The system administrator invoked subsequent `make_net_recovery` archive operations either from the Ignite-UX interface or from the command line of the client. (Steps 1 through 7 are required only for the first archive creation.)

After the disaster occurred, and the point of failure was identified, the replacement disk hardware was ordered and received. The system administrator started recovery operations using the following steps:

1. The administrator shut down and powered off the client system, and replaced the failed boot disk.
2. Since the new disk had no operating system, the administrator powered on the client and brought it to the firmware prompt, initiating remote boot operations to pull an install kernel from the Ignite-UX server.
3. Using the Ignite-UX TUI on the client console, the administrator selected an archive for recovery. (Archives are labeled based on the description provided when the archive was created. The default naming convention indicates the creation date and time, such as `Recovery Archive 2006-02-04,07:43`).
4. The system administrator poured a cup of coffee and monitored the recovery until it completed, which was about 45 minutes later.

Command-Line Examples

The following `make_net_recovery` examples assume that the *defaults* file was automatically generated through the “Add New Client for Recovery” option, with a client name of *hal*. The hostname of the Ignite-UX server is *ibanez*.

```
ibanez: /var/opt/ignite/clients/hal/recovery # cat defaults
#
# This file is used to specify defaults for make_net_recovery.
# It is overwritten every time the Recovery UI is run,
# even if an archive is not created.  It should not be used
# as a reliable source of information about any given archive.
#
RECOVERY_TYPE=net
RECOVERY_LOCATION=ibanez:/var/opt/ignite/recovery/archives/hal
TAPE_DESTINATION=none
RECOVERY_DESCRIPTION=Recovery Archive
SAVE_NUM_ARCHIVES=2
ARCHIVE_TYPE=tar
```

Since the Ignite-UX server already has an archive configuration, the following command may be run on the client’s command line (as opposed to being initiated through the Ignite-UX server interface) to create a network recovery archive that uses the existing client configuration on the *ibanez* server.

```
# make_net_recovery -s ibanez
```

A `cron` job can be defined to launch this command automatically at desired intervals.

The following command creates a local tape recovery archive of the system on which it’s run, using settings from the last run of this command on this machine:

```
# make_tape_recovery -s ibanez
```

Suppose that the */var* filesystem on the Ignite-UX server is close to capacity and that there is insufficient disk space to store a new archive. The following command overrides the existing `recovery_location` parameter, assuming that the new archive location has been manually created and properly exported via NFS.

```
# make_net_recovery -s ibanez -a ibanez:/depot/recovery/archives/hal
```

These three examples each create a minimal recovery archive, containing only the directories and files that Ignite-UX considers essential for a functional system as defined by the *mnr_essentials* file.

> **Tip ✎**
>
> Typically, symbolic links are not followed during the creation of an archive; only the link is backed up. However, if a link is listed in *mnr_essentials*, the link is followed to the file it links to, and that file is backed up.

The following command examples use the `-x inc_entire=` option to create an archive that contains the entire root volume group (*vg00* in this example), including as part of the root volume group any directories and files listed in *mnr_essentials*, no matter what volume group they reside in:

```
# make_net_recovery -x inc_entire=vg00 -s ibanez
# make_tape_recovery -x inc_entire=vg00
#make_tape_recovery -x inc_entire=vg00 -s ibanez
```

> **Warning ☠**
>
> The –A option of the `make_tape_recovery` and `make_net_recovery` commands is quite different from the –A option to the obsolete `make_recovery` command. –A now means that if any additionally included file or directory resides outside of the root volume group, disk group, or disk, that entire volume group, disk group, or disk is included in the archive.

Verifying Archive Contents

You should test system recovery archives before relying on them to restore a production system. If a system recovery archive cannot be restored, it is better to discover this before the failure of a production system. The only foolproof method of verifying a recovery image is by performing a full recovery of that image back onto the system where it originated. Here are some ways to test that Ignite-UX is doing what it's supposed to:

Use preview mode to determine whether an archive of the currently running system will succeed.

You can specify the –p argument to `make_net_recovery` or `make_tape_recovery` to preview the processing that takes place without actually creating the archive. The following example was run from a client system, specifying archive preview mode on *ibanez*.

```
# make_net_recovery -s ibanez -p
```

Inspect the *flist* file on the Ignite-UX server to view the list of files that would be part of an actual archive. If `make_net_recovery` preview-mode sessions are run in succession before actually creating an archive, Ignite-UX creates new archive recovery directories, each using a date and timestamp, with the most recent directory having a link created to it that is called *latest*.

```
# view /var/opt/ignite/clients/hal/recovery/2006-06-22,12:01/flist
```

After running `make_net_recovery` in preview mode, subsequent execution using the `-r` option continues with the creation of an archive, using the configuration information in the directory that is pointed to by the latest link.

If you run the `make_tape_recovery` command in preview mode, review the *flist* file on the client at the following location. Successive `make_tape_recovery` preview sessions followed by one with the `-r` option operate in a manner similar to that of `make_net_recovery`.

```
# view /var/opt/ignite/recovery/latest/flist
```

Review the list of directories and files in the recovery archive.

The *flist* file is generated in preview mode when the archive is actually created. Read *flist* in a text editor.

Determine whether you can read the archives.

The following commands read the `tar` index information from the specified recovery archive stored on the disk of an Ignite-UX server and write that information to the specified output file. `tar` is the default archive format (but you can use `cpio` instead). The `pax` command can read either format. The following examples use `tar` and `pax` to access one of two `make_net_recovery` archives stored under unique directory names that are based on the creation date and time of each archive:

```
# cd /var/opt/ignite/recovery/archives/hal
# file 2006*
2006-01-24,19:51:       gzip compressed
2006-01-31,17:06:       gzip compressed
# gunzip -c 2006-01-31,17:06 | tar -tvf - > /tmp/file.list
#gunzip -c 2006-01-31,17:06 | pax -vf - > /tmp/file.list
```

You can use a similar approach for tapes produced using `make_tape_recovery` on HP9000 systems. This example reads from a tape loaded into the specified drive (note the use of a no-rewind device file), seeking past the boot components at the beginning of the tape to access the archive. `pax` reads the `tar` index information from the recovery archive and writes it to the specified output file.

```
# mt -r /dev/rmt/0m rew
# mt -t /dev/rmt/0mn fsf 1
#pax -vf /dev/rmt/0m > /tmp/file.list
```

Compare the contents of an existing recovery archive against the system on which it was created using either the check_net_recovery or check_tape_recovery command.

If the archive is stored on a remote server, run the following command on the client system to be verified, specifying the hostname of the Ignite-UX server that contains the network recovery archive. The output of this command reports whether any files in the archive have been added, deleted, or changed since the archive was created.

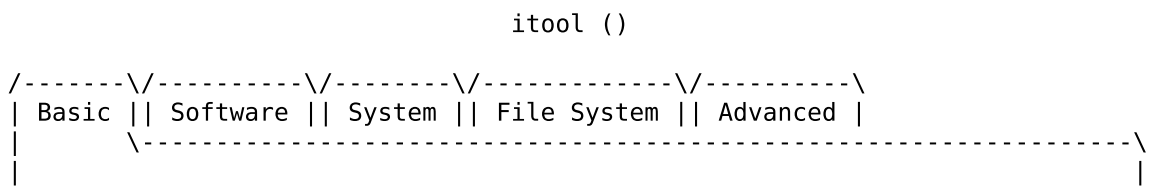
```
# check_net_recovery -s ibanez
```

Verify that the client system will boot from the Ignite-UX server containing its make_net_recovery archive (or from the recovery tape that was created with make_tape_recovery).

For the latest information on recovery tape operations for HP Integrity systems, refer to documentation found online at http://docs.hp.com. Use a current version of Ignite-UX to make full use of commands such as `make_ipf_tape` that enable new tape functionality on HP Integrity systems. Support for native UEFI 2.0 bootable tapes starts with Ignite-UX version C.6.8. (This requires firmware updates on most HP Integrity servers to enable native tape boot.)

For HP Integrity systems, verify that the attached tape drive and host bus adapter (HBA) supports direct boot capabilities for tapes created in bootable UEFI 2.0 format. For tapes that are formatted for booting on an HP9000 system, use a dual-media boot process on HP Integrity systems. You first boot from a DVD (such as the OS install media) and then select the Logical Interchange Format (LIF) formatted tape for booting. All recovery tapes created before Ignite-UX version C.6.3 are HP9000 format tapes. Recovery tapes created using Ignite-UX versions C.6.3 to C.6.7 are not UEFI 2.0-compliant; with these you must boot from DVD first, then tape.

After verifying that a client system may be booted for archive recovery, use the Ignite-UX `itool` interface on the client console to review the configuration that would be installed were the recovery operation to actually be initiated:



```
  | Configurations:  [ HP-UX B.11.00 Default    ->] [ Description...  ]       |
  |                                                                           |
  | Environments:    [                            ->] (HP-UX B.11.00)         |
  |                                                                           |
  | [ Root Disk...  ] SEAGATE_ST32151N, 8/16/5.5.0, 2048 MB                   |
  |                                                                           |
  | File System:     [ Logical Volume Manager (LVM) with VxFS  ->]            |
  |                                                                           |
  | [ Root Swap (MB)... ] 1024      Physical Memory (RAM) =  2560 MB          |
  |                                                                           |
  | [  Languages...   ]                  [ Keyboards...  ] [ Additional...  ]|
  |                                                                           |
  \---------------------------------------------------------------------------/
       [ Show Summary...  ]                          [ Reset Configuration ]
 ----------------------------------------------------------------------------
  [  Go!   ]                       [ Cancel ]                       [  Help  ]
```

Troubleshooting Recovery Operations

Generally speaking, initial troubleshooting includes the following questions.

- Has the Ignite-UX server in question been used successfully in other archive creation and recovery operations? If so, what is different about the current operation? If not, has the Ignite-UX server been properly configured?
- Are the Ignite-UX client and server systems on the same network subnet? If not, has an Ignite-UX boot helper server been configured?
- Was this particular recovery archive created on the same physical system now being recovered? Has this archive ever been successfully used in a recovery operation?
- What specific error messages have been observed? What details are found in related logfiles?

When a recovery operation fails, look for a message on the client console that explains the nature of the problem, along with a prompt to work from a debug shell. For example:

```
ERROR:   The ifconfig(1M) command (ifconfig lan2 inet 15.43.233.125 netm ask
         255.255.248.0 up) failed to enable the lan2 interface (return value:
         1).  Check that the system's networking hardware is functioning and
         properly connected.
ERROR:   Could not start networking.
         The configuration process has incurred an error, would you like
         to push a shell for debugging purposes? (y/[n]): y
       * Type "exit" to reboot, "exit 2" to ignore error
To continue the recovery after working from the debug shell, enter exit 2.
```

In another example, after an archive recovery operation was initiated, the network-based Ignite-UX server monitored the client's *install.log* at defined intervals, eventually reporting the following warning.

```
WARNING: Client hal appears to be hung.  Since the server started
         checking, the client's logfile has not been updated in 20 minutes.
         The timeout in effect is 20 minutes.
```

After seeing the warning, review the client's *install.log* to determine the point in the recovery process where a failure may have occurred. In this example, the client name is *hal*, and the client's install log would be at */var/opt/ignite/clients/hal/install.log* on the Ignite-UX server. On the client itself, from the debug shell, you can view the install log at */var/opt/ignite/local/install.log*:

```
* Loading_software:  Begin
       * Installing boot area on disk.
       * Enabling swap areas.
       * Backing up LVM configuration for "vg00".
       * Processing the archive source (Recovery Archive).
       * Fri Sep 20 15:08:07 EDT 2002: Starting archive load of the source
         (Recovery Archive).
ERROR:   Cannot load OS archive (Recovery Archive)
         The configuration process has incurred an error, would you like
         to push a shell for debugging purposes? (y/[n]):        *
```

This error could have a number of possible causes, including a physical network connection that is failing or otherwise performing poorly. In the case of poor performance, determine whether a network switch is being used between the Ignite-UX client and server. While booted to an Ignite-UX install kernel, if the client's network interface fails to auto-negotiate the network speed and duplex settings with the switch, the default is 100/half-duplex—regardless of the settings on the switch. This results in an extremely slow recovery archive installation.

If auto-negotiation is not used on the switch being used for the recovery operation, and a mismatch in network speed and duplex settings occurs, the default settings used by Ignite-UX may be modified in the installation filesystem (*INSTALLFS*) located on either the Ignite-UX server or boot helper server being accessed by the client system for remote booting. For this change, the `_hp_lanadmin_args` parameter is modified within the Ignite-UX *INSTALLFS* through the use of the `instl_adm` command as described here. For reference, the Ignite-UX configuration file syntax is discussed in the `instl_adm(4)` manpage.

First, read the current Ignite-UX configuration into a temporary file using the following command:

```
# instl_adm -d >/tmp/installfs.cfg
```

Next, as detailed in the `instl_adm(4)` manpage, set `_hp_lanadmin_args` equal to `-X 100FD` in the temporary file */tmp/installfs.cfg*.

Finally, write the new Ignite-UX configuration to *INSTALLFS*.

```
# instl_adm -f /tmp/installfs.cfg
```

System Cloning

In cloning, the system recovery archive is extracted onto a different physical computer from the computer that was used to generate the archive. This is the norm in many recovery scenarios. Cloning is also becoming popular with system administrators who seek shortcuts to managing systems with similar configurations.

When an archive is created from one system, many aspects of that system’s configuration may be similar to those of other systems. For example, the systems may be on the same subnet with the same route tables. Or they may have the same network, SCSI, or Fibre Channel adaptor cards. Perhaps they even have the same model of disk arrays or tape libraries—all running the same versions of various firmware. Yet there is still the potential for incompatibilities when using recovery archives to clone systems. The recovery process may fail if the system configuration stored in the archive is too dissimilar from the new target system.

When you attempt to restore an archive onto a system with different hardware, the system recovery tool should detect dissimilar hardware. This detection can normally be noticed at the system console during recovery. If the cloning process completes, it may become necessary to change system settings such as IP addresses or hostnames. This can be done post deployment, but these particular parameters can be modified prior to archive extraction using the Ignite-UX `itool` interface on the client console.

Consider an example in which a system targeted for archive recovery has a root volume group with different sized disks/LUNs or different numbers of disks, from those that were on the system that created the archive. Ignite-UX detects that the client system has a different hardware

configuration and brings up the `itool` user interface on the client, from where the system administrator may reconfigure the related installation parameters concerning disks, volumes, and filesystems. If the system administrator does not manually reconfigure them, Ignite-UX attempts to modify logical volume and filesystem sizes during installation to ensure that there is, by default, 10 percent free space. If there is insufficient disk space for Ignite-UX to work with, it generates an error.

Security

It is important to understand the security implications of using the HP-UX system recovery tools. When setting up the Ignite-UX server, certain services may be required to run, such as NFS, TFTP, and BOOTP, as well as optional commands such as `bootsys`. You should evaluate the security policies of the organization prior to setting up these recovery tools and examine the risk factors associated with running these protocols and commands. You should also review the network environment. Will firewalls be crossed, and what restrictions will be imposed?

You can disable some of these services and commands when they are not being used. For example, the `bootsys` command that initiates remote booting of a client from the Ignite-UX server may be blocked on specified servers (refer to the `bootsys` manpage). Or if the organization has a security policy stating that NFS cannot be used on any server, consider running `make_tape_recovery` on each local client instead of `make_net_recovery` (or disabling NFS after creating each `make_net_recovery` archive).

Remember that system recovery typically creates two NFS mount points: one for the directory on the Ignite-UX server containing client configuration information and another for the client archive. Also recall that the Ignite-UX server is not necessarily the same machine as the archive server. When the archive server is different from the Ignite-UX server, the */etc/exports* file should be modified to allow access only to the client(s) being archived.

An Ignite-UX server may be configured to disallow access by anonymous clients—for example, by providing temporary IP addresses for installation and booting only to

certain network interfaces that are identified by link-level hardware MAC addresses.

Consider the soundness of the disaster recovery planning and implementation. A single system may function as both an Ignite-UX client and server. But if the server fails, from where will it be recovered? If the environment is compromised, then how, and from where, will it be rebuilt? Even where two or more Ignite-UX servers are in use across a network and in locations remote to one another, be sure to have a recovery plan for each recovery server.

It is also important to ensure the integrity and security of the recovery archives themselves. Once an archive is created, the entire system is captured in a file or on a tape media and contains just about everything that an attacker needs to exploit a system or an entire network. For example, there are several password cracking programs that can be used to guess the passwords found in a system's */etc/passwd* file. Other critical information such as route tables can be obtained from the archive as well. Are the servers and tape libraries physically secure? Is there off-site storage of tape media, and are those remote sites secure? Are proper Unix security practices being followed on the Ignite-UX server?

At the time of this writing, the system recovery archives are unencrypted. This means that the physical security of system recovery images, whether on tape or stored on a network server, is essential.

System Recovery and Disk Mirroring

The archive created by system recovery tools requires some additional steps to preserve mirrored disk configurations. Software-level disk mirroring is used by volume managers to provide redundancy of disk and filesystem resources. The data from one disk is mapped to another disk of the same size (in LVM this occurs at the logical volume level). If the one disk is no longer operational, the system remains functional based on failover mechanisms to the active disk. When the recovery archive is restored, the system recovery tools do not preserve the software-level mirroring configurations.

In the Ignite-UX environment, system administrators typically have scripts that reconfigure the software-level mirroring after the system has been installed from the recovery archive. Ignite-UX has its own configuration language, enabling custom scripts to be invoked automatically as post-configuration steps. These scripts do not exist unless the system administrator creates and then integrates them into the system recovery configuration. The following document is installed with Ignite-UX and provides instructions for modifying configuration files to restore disk mirrors:

```
/opt/ignite/share/docs/diskmirror.pdf
```

The tools covered in this chapter can be invaluable in ensuring that you can recover your HP-UX systems if their operating system disks are damaged. We hope our information proves helpful.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 13. AIX Bare-Metal Recovery

This chapter explains the procedure you would use to recover your IBM AIX operating system disk in case of a complete system failure—when you are left with nothing but bare metal.

> **Tip ✎**
>
> This chapter was contributed by Mark Perino of Hitachi Data Systems. Mark tackles performance, AIX, Linux, and NAS problems when he's not writing, hiking, or snowshoeing in Yosemite with his wife, Marissa, and kids, Steven and River.

IBM was the first Unix vendor to deliver a true bare-metal recovery tool. The `mksysb` command makes a complete "bootable" backup of the root volume group (`rootvg`) only. This allows you to perform a bare-metal recovery of an AIX host's operating system. You can even back up using `mksysb` to a bootable tape, CD/DVD, or a Network Install Manager (NIM) server.

Unix Backup & Recovery mentioned using `Sysback` to make volume group backups. The `Sysback` utility has been incorporated into some commercial products from IBM and other vendors with the release of AIX 5.X. This book's primary focus is on free methods to back up and restore, so we will not be covering `Sysback`.

IBM's mksysb and savevg Utilities

The basis for a bare-metal recovery of an AIX system is the `mksysb` utility, which is included in AIX. It backs up all the

files in the root volume group. `mksysb` backs up the `rootvg`, including:

- Boot files
- Base operating system (BOS)
- System and configuration files in the `rootvg`
- Additional software installed in the `rootvg`

It also backs up:

- `rootvg` volume group information
- Logical volume information
- Paging space information

It does not consume space in the backup to save the paging space; it recreates it on a restore. `mksysb` is primarily useful for a bare-metal recovery. It also has limitations that may prevent it from becoming your only backup solution. Here are some of those limitations:

- It cannot back up filesystems on something other than `rootvg` (see `savevg`)
- It cannot back up raw logical devices
- It has a limited ability to preserve logical volume layout (AIX 4.x)
- It has restrictions restoring across different RS/6000 architectures (see the later section “System Cloning”)
- It cannot track backups or perform incremental backups
- It does not have tape robot controls
- It can back up to a remotely attached tape drive but cannot remotely boot from it

Before AIX 5.x, there were significant differences in the `mksysb` program from one version to the next. As of AIX 5.x, `mksysb` is more stable. All AIX 5.x versions can preserve logical volume characteristics and paging space size. `mksysb` does not back up raw logical volumes. It cannot back up anything other than the root volume group.

`mksysb` can also be a good solution if you need to make occasional tape backups of your system in case of disk failure. It also can make an excellent companion to other backup programs that handle your application and user data, and provide things that `mksysb` cannot, such as incremental backups and remote restores. `mksysb` is not a good solution for environments that are using raw logical volumes; have data outside `rootvg`; or need to do incremental backups, flexible restores, and backups across the network.

You should always make backups before and after applying IBM maintenance-level upgrades. Maintenance-level upgrades with patches for JFS and JFS2 may not be backward-compatible. So without current and older backups, you run the risk of not being able to mount your filesystems.

Versions 4.3.3 and 5.x of AIX include a utility called `savevg` , which is actually a link to the `mksysb` command. When invoked this way, you can back up any volume group on your system, but the other `mksysb` constraints still apply.

Here are other requirements of the `savevg` command:

- Volumes backed up using the `savevg` command must be varied on and have their filesystems mounted.
- `savevg` cannot back up remotely mounted NFS and CIFS filesystems.
- Filesystems must be of type JFS or JFS2.

`mksysb` and `savevg` can back up to different types of media:

- Tape (bootable in the case of `rootvg`)
- CD/DVD (bootable in the case of `rootvg`)
- Image file on NFS mount

`mksysb` and `savevg` can be restored from:

- Tape (bootable in the case of `rootvg`)
- CD/DVD (bootable in the case of `rootvg`)
- Internal disk (not recommended for `rootvg`)
- NFS mount (bootable when used with a NIM server)

mksysb and savevg Format

`mksysb` works by creating several files on its backup tape, CD, DVD, or image file. The first few files boot the kernel and provide the LVM layout of the data that was backed up. The last files contain the files to be restored in `backup` format for AIX 4.3.3 and 5.x. (The AIX `backup` format is essentially the same as `dump`.) `savevg` backups are essentially the same except there is no bootable kernel information. Figure 13-1 shows a logical representation of such a tape.



Figure 13-1. A logical representation of a mksysb tape

The tape block size for the first three files is 512 bytes while the block size for the image data is variable, based on the settings of the tape drive at the time of the backup. If you choose 0 as the block size for that device, systems will,

when possible, back up with 1024 bytes. When backing up to CD/DVD or image file, it is not necessary to specify a block size.

Preparing to Use mksysb and savevg

There are three common types of `mksysb` backups:

- Backing up and restoring from a tape drive
- Creating a bootable CD/DVD
- Backing up to an NFS mount and restoring via NIM

The first two require physical media. Both tape and CD/DVD produce bootable restore and recovery images. Backing up to an NFS mount is limited by the speed of the IP network when doing backups. Restore speed for NIM is limited by the speed of the network as well. Backing up to an NFS share and restoring via NIM is more work to set up but can be automated using scripts to make periodic backups of the `rootvg` without being dependant on a human to load media.

Another option that some people script is backups to a local disk that is not a member of `rootug`, and then a script to move them off to a remote host using FTP/SCP/RCP. Most users should avoid doing this. It adds a potentially weak link to your backup strategy. What happens when the copy script fails, and the backup you need is good but physically on a failed machine? A simple restore just got a whole lot harder. If NFS just isn't an option, and you must use another transfer method, be sure to write solid verification scripts both on the client and the destination server. Whatever you do, don't ever send your `mksysb` backups to the `rootvg`.

If you find yourself in this situation, you can restore from an older `mksysb` or CD-ROM, apply maintenance-level releases to get to the same level as when the host failed, mount the backup filesystem, and perform a second restore from the desired `mksysb`.

After you decide where to back up to, you need to decide whether you are going to quiesce the system to take a root backup. Will you log off all users and/or shut down your applications for the duration of the backup? The answer depends on your system configuration. The backup will likely take at least an hour, and any changes to files during the backup may lead to an inconsistent image on tape. There should not be any problems backing up files that are currently open, although any pending writes to the file at the time it is backed up will obviously not make it to the tape archive. If you have most or all of your user data off `rootvg`, and your system files are not constantly changing, you should have no problem doing `mksysb` backups on a "live" system.

If you are using this `mksysb` image to build other systems, you may want to prepare the `rootvg` even further. For example, you may want to do the following:

- Clear out */tmp*.
- Disable the network (by commenting out from *inittab*, *rc.net*, *rc.tcpip*, and *rc.nfs* anything that hangs the computer if it has no network or causes an IP address conflict).
- Remove the root password or set it to a well known password in your environment.
- Clear the error report using `errclear`, and clear */var/spool*, */var/adm*, */var/logs*, and the like.

Next, decide what needs to be backed up. You can prevent specific files from being backed up to the archive by creating a file called */etc/exclude.rootvg* and using the `-e` flag to `mksysb` (or `savevg`). Entries in this file can be simple lists of files, such as:

```
/etc/passwd
```

If you use the `-e` flag, `mksysb` filters out the files in your */etc/exclude.rootvg* using `egrep`, so in effect, it supports `egrep`'s syntax. This allows complex entries such as:

```
.*/core$
^/tmp/
.*\.o$
```

Preparing for the restore

The */image.data* file in the `/` filesystem being backed up contains information about how disks, filesystems, logical volumes, and paging space are rebuilt when the backup is restored. In general, the entries in *image.data* correlate to flags in the `mklv` and `crfs` commands (commands used to create logical volumes and filesystems). Running the `mkszfile` command before the `mksysb` creates the *image.data* file itself. You also can start `mksysb` with the `-i` option, which tells `mkszfile` to generate the *image.data* file automatically. Running `mkszfile` independently enables you to edit *image.data* and change what is backed up, the size of the filesystems, and other variables. You also can edit *bosinst.data* to customize what runs after the image is restored. If you run `mkszfile` and edit *image.data*, make sure you run `mksysb` without the `-i` option, or your modifications will be overwritten.

One useful option to edit in *bosinst.data* is the `RECOVER_DEVICES` variable. This specifies whether or not to restore the ODM definitions that were in place at the time

of the backup. The default (YES) is correct if you plan to restore to the same or similar hardware. If you are performing a backup for a generic system image, set it to NO.

If you specify the -m option to mkszfile, a map file is created for each logical volume in the archive. Each map describes where on the disk the logical volume should be created at restore time. This is similar to the options available to the mklv command (exact layout ability). The map file for a given logical volume contains one line for each physical partition (PP) it occupies, along with the hard disk (hdisk) it is on. Note that this is useful only if the target system has exactly the same disk layout as the source system.

Here is an example of what the *image.data* file contains. The AIX documentation gives a more complete description of all the options contained in this file.

```
[SHRINK = yes]
```

Shrinks the filesystems on restore

```
[VG_SOURCE_DISK_LIST = hdisk0]
```

Changes the target disk for the restore

Once backed up, these files cannot be edited on tape or CD/DVD media. You can, however, create a customized diskette containing an *image.data* and/or *bosinst.data* file. The diskette could be used at restore time instead of the files in the backup. To create a customized diskette, follow this simple procedure.

Edit *image.data* and/or *bosinst.data* extracted from a tape or other media (see the instructions for extracting a single file from a mksysb tape), place a diskette in the drive, and issue the following command:

```
# ls ./bosinst.data ./image.data | backup -ivqf /dev/rfd0
```

Backing Up with mksysb

`mksysb` has several options, which are described in this section. First, you can back up and restore from a locally attached tape. You can also back up to a remote tape drive, but you must restore from a locally attached tape drive. Next, you can use `mksysb` to make a system backup on disk, such as a centrally located NFS filesystem. If you do that, you have two recovery choices. The first choice is to create bootable DVDs/CDs using `mkcd` and then boot from those during a recovery. The second recovery choice is to boot from a NIM and restore directly from the NFS-mounted image. Finally, you can also create DVDs/CDs directly using `mkcd`. Let's take a look at these options.

mksysb Summary

Here is a quick summary of the options of `mksysb` that I'll be using in this section:

`-e`

Excludes files listed in the */etc/exclude.rootvg* file from being backed up. The rules for exclusion follow the pattern-matching rules of the `grep` command.

`-i`

Calls the `mkszfile` command, which generates the */image.data* file. The */image. data* file contains information on volume groups, logical volumes, filesystems, paging space, and physical volumes. This information is included in the backup for future use by the installation process.

`-m`

Calls the `mkszfile` command with the `-m` flag to generate map files. (Using the `-m` flag causes the functions of the –`i` flag to be executed also.)

Backing Up rootvg to Locally Attached Tape

The simplest scenario is to back up all filesystems in `rootvg` with a locally attached tape drive. In this situation, issue the following command from a root prompt:

```
# mksysb -i /dev/rmt0
```

The difference between the various ways to access a tape device on AIX systems is illustrated in Table 13-1 and explained in detail in the manpages.

Table 13-1. AIX device-naming conventions

Device name	Density	Rewind on close	Retention on open
*/dev/rmt**	Setting #1	Yes	No
/dev/rmt.1*	Setting #1	No	No
/dev/rmt.2*	Setting #1	Yes	Yes
/dev/rmt.3*	Setting #1	No	Yes
/dev/rmt.4*	Setting #2	Yes	No
/dev/rmt.5*	Setting #2	No	No
/dev/rmt.6*	Setting #2	Yes	Yes
/dev/rmt.7*	Setting #2	No	Yes

This can also be achieved through the System Management Interface Tool (`smit`) menus, of course, and the following fast path brings you directly to the correct screen:

```
# smit mksysb
```

You should be able to use any locally attached tape drive supported by AIX.

Backing Up rootvg to a Remote Tape Drive

You can perform a `mksysb` directly to a remote tape by following this procedure. For the purpose of this example, consider two machines:

tapeserver

The machine with the local tape drive

client

The machine to be backed up

First, make sure the */.rhosts* or */etc/hosts.equiv* file is set up properly on both machines to allow `rsh` from the client to the tapeserver. On the tapeserver, use this procedure to manually create a `mksysb` image:

1. Set the default block size of the tape to 512 for the boot section:
2. `tapeserver# chdev -a block_size=512 device`
3. Rewind the tape:
4. `tapeserver# mt -f device rewind`
5. Create the boot image files:
6. `tapeserver# bosboot -d no-rewind-device -a`
7. Write the boot image to the tape:
8. `tapeserver# mkinsttape no-rewind-device`
9. Write a dummy table of contents:
10. `tapeserver# echo "Dummy tape TOC" | dd of= no-rewind-device bs=512 conv=sync`

11. Change the block size to 1024 for the rest of the tape:

12. ```
    tapeserver# chdev -a block_size=1024
                                device
    ```

Now that we have the bootable part of the tape (i.e., the boot block), the installation files, and the dummy table of contents, we need to transfer the data. On the client, run `mkszfile` , and create a file called */etc/exclude.vg* that contains only */tmp/mksysb. pipe*. Then, direct `mksysb` to a named pipe. Finally, `cat` the contents of the pipe to the tapeserver's tape drive:

1. Create a pipe for `mksysb`:

2. ```
   client# mknod p /tmp/mksysb.pipe
   ```

3. `cat` the pipe to the remote tape drive:

4. ```
   client# cat /tmp/mksysb.pipe \
                                   | rsh
                               tapeserver
                               "dd of=
                               device
                               obs=100b bs=1024 > /dev/null 2>&1" &
   ```

5. Run `mksysb` with the `-e` argument against the previously created pipe:

6. ```
   client# mksysb -e /tmp/mksysb.pipe
   ```

If you are lazy, a script written by Henk van Doorn called `rmksysb` does all of this.

Backing Up to Disk

It is possible to back up a system to a locally attached disk, but that obviously presents challenges during a restore. It's even possible to back up the `rootvg` to a file in the `rootvg`, but that would really be a bad idea for a few different

reasons. A common practice is therefore to set up a NFS share on a separate server to hold `mksysb` backups.

The filesystem must be large-file-enabled, and you must set `ulimit` to allow 9 GB files (by default, `ulimit` is set to 1 GB).

To see the current setting of `ulimit`:

```
Client # ulimit -f
2097151
```

The systemwide default setting is configured in */etc/security/limits*. You can also change it in a shell using the `ulimit` command (9 GB plus a little for overhead):

```
Client # ulimit -f 19000000
```

Now that we can write out a large file (over 1 GB), we can write a backup of the `rootvg` to an NFS mount on the client system. In the following examples, we use */ NFS_mount/mksysb* from the server *NFS_Server* to hold `mksysb` backups. We'll use the command `mksysb –e –m /directory/filename`. The `–e` option tells it to exclude patterns listed in */ etc/exclude.rootvg*, and the `–m` option specifies the media to write to. The last argument should be the full pathname of the file in which to store the image.

```
Client # mksysb -e -m /NFS_mount/mksysb/client-6-1-06.msb_image
Creating information file (/image.data) for rootvg..
Creating list of files to back up.
Backing up 39932 files.........#
# 39932 of 39932 files (100%)
0512-038 mksysb: Backup Completed Successfully.
```

It's just that easy to make a backup of the `rootvg`. We now have a backup that we can restore to this host. In the event of a minor problem, we can restore a single file, and in the event of a major problem, we can create a bootable restore DVD or restore using NIM.

Making a Bootable DVD/CD from an Existing mksysb

Once you create a `mksysb` image on disk, you can use it to create a set of bootable DVDs or CDs that can be used to restore the system very quickly. This process has three steps:

1. `mkcd` creates temporary files that it uses to create the ISO images.
2. `mkcd` creates ISO images from the temporary files.
3. The ISO images are then used to burn the actual CDs/DVDs.

We are going to need some local scratch space for the first two steps. This scratch space should not be in `rootvg` and needs to have enough space to hold the temporary CD/DVD filesystem and the CD/DVD ISO images. For each CD, this is about 645 MB, and each DVD holds 4.38 GB. The temporary files can probably be on an NFS directory, but it's probably best if the ISO images are stored locally to prevent buffering problems when creating the DVDs or CDs.

We'll use the following arguments to the `mkcd` command to perform this task:

`-e`

Excludes the files and/or directories from the backup image listed in the */etc/exclude.volume_group* file. You cannot use this flag with the `-m` or –`s` flags.

`-m` *`filename`*

`mksysb_image` specifies a previously created `mksysb` image. If you do not give the `- m` flag, `mkcd` calls `mksysb`.

(See the -M flag for more information about where the mksysb image is placed.)

-C *directory*

Specifies the filesystem used to create the CD filesystem structure, which must have at least 645 MB of available disk space (up to 4.38 GB for DVD-sized images). The CD image consumes only as much room as necessary to contain all the data on the CD.

-I *directory*

Specifies the directory or filesystem where the final CD images are stored before writing to the CD-R, DVD-R, or DVD-RAM device. If this flag is not used, mkcd uses the */mkcd/cd_images* directory if it already exists. If it doesn't exist, the command creates the */mkcd/cd_images* filesystem in the volume group given with the -V flag, or in rootvg if that flag is not used.

-P

Creates physical partition mapping during mksysb or savevg creation. You cannot use this flag with -m or -s flags.

-R

Prevents mkcd from removing the final CD images. mkcd defaults by removing everything that it creates when it finishes executing. The -R flag allows multiple CD image sets to be stored or for CD creation (burn) to occur on another system. If multiple volumes are needed, the final images are uniquely named using the process ID and volume suffixes.

-S

Stops mkcd before writing to the CD-R, DVD-R or DVD-RAM without removing the final CD images. The -S flag

allows multiple CD sets to be created or for CDs to be created on another system. The images remain in the directory marked by the `-I` flag or in the */mkcd/cd_images* directory if the `-I` flag is not used. If multiple volumes are required, the final images are uniquely named using the process ID and volume suffixes.

`-d` *`CD-or-DVD-writer-device`*

Indicates the CD-R, DVD-R, or DVD-RAM device (*/dev/cd1*, for instance). This flag is required unless you use the `-S` flag.

In the following examples, the host named *NFS_Server* has a DVD writer. It also has a local 18 GB filesystem called */mkcd* that we use for both the temporary images as well as the ISO images. *NFS_Server* also shares the filesystem */NFS_Share/mksysb* with other clients so they can write their `mksysb` backups there. (This is the same directory we backed up to in the previous example.)

The example that follows uses the NFS server to create a bootable CD image from the `mksysb` image we created on the NFS mount in the previous example (*/NFS_share/mksysb/client-6-1-06.msb_image*). We use the directory */mkcd/cd_fs* to hold the temporary images and the directory */mkcd/cd_images* to hold the final ISO images.

```
NFS_Server # mkcd -m /NFS_share/mksysb/client-6-1-06.msb_image -C /mkcd/cd_fs
-I /mkcd/cd_images -R –S
Initializing mkcd log: /var/adm/ras/mkcd.log...
Verifying command parameters...
Populating the CD or DVD file system...
Copying backup to the CD or DVD file system...
.
Building chrp boot image...
Creating Rock Ridge format image:
/mkcd/cd_images/cd_image_295108.vol1
Running mkisofs ...
.
mkrr_fs was successful.
```

```
Making the CD or DVD image bootable...

Copying the remainder of the backup to the CD or DVD file
system...
Creating Rock Ridge format image:
/mkcd/cd_images/cd_image_295108.vol2
Running mkisofs ...
.
mkrr_fs was successful.


Copying the remainder of the backup to the CD or DVD file
system...
Creating Rock Ridge format image:
/mkcd/cd_images/cd_image_295108.vol3
Running mkisofs ...
.
mkrr_fs was successful.
```

This gives us three ISO CD images in */mkcd/cd_images*. The temporary data that was copied to */mkcd/cd_fs* was removed after the cd image was created. We can now transfer that image to another server with a CD drive and burn the *.ISO* images.

If we want to create a DVD image, we add the –L flag to the command:

```
NFS_Server # mkcd –L -m /NFS_share/mksysb/client-6-1-06.msb_image
-C /mkcd/cd_fs -I /mkcd/cd_images -R –S
Initializing mkcd log: /var/adm/ras/mkcd.log...
Verifying command parameters...
Populating the CD or DVD file system...
Copying backup to the CD or DVD file system...
.......
Building chrp boot image...
Creating Rock Ridge format image: /mkcd/cd_images/cd_image_405642
Running mkisofs ...
........
mkrr_fs was successful.

Making the CD or DVD image bootable...
```

Our earlier example created three ISO images of CD size, but in this example, we specified the –L flag, so we have

one large DVD ISO image. You can use these ISO images to create a CD or DVD on any host.

If you want to create the images *and* burn the CD/DVD all in one step, you can drop the –S and –R options and add the –d option, followed by the device name of the CD/DVD burner:

```
NFS_Server # mkcd –d /dev/cd0 –e -m /NFS_share/mksysb/client-6-1-06.msb_image
-C /mkcd/cd_fs -I /mkcd/cd_images
```

If you want to burn a CD from an ISO image created earlier, use a command like this:

```
NFS_Server # burn_cd /dev/cd0 /NFS_share/mksysb/client-6-1-06.msb_image
```

If there is more than one CD image file, repeat the command for each image file.

If you want to burn a DVD from an ISO image, add the –d flag:

```
NFS_Server # burn_cd –d /dev/cd0 /NFS_share/mksysb/client-6-1-06.msb_image
```

Creating a CD/DVD Backup in One Step

If you are lucky enough to have a CD/DVD writer available on the system you want to make DVDs for and have sufficient local scratch space to build a CD image, you can use `mkcd` to create the CD/DVD in one step. Although all of the steps in the previous method are still required and performed, `mkcd` manages and automates the whole process, from making the `mksysb` backup all the way to burning the actual DVDs/CDs. In the following example, we want to make a bootable DVD-sized backup of the `rootvg` with map files (restore LVM), excluding all files and directives in */etc/exclude.rootvg*:

```
NFS_Server # mkcd -L -P –e –M /NFS_Share/mksysb –C /mkcd/cd_fs
–I /mkcd/cd_images -R –S
Initializing mkcd log: /var/adm/ras/mkcd.log...
```

```
Verifying command parameters...
Creating image.data file...
Creating mksysb image...

Creating list of files to back up.
Backing up 39945 files..............
39945 of 39945 files (100%)
0512-038 mksysb: Backup Completed Successfully.
Populating the CD or DVD file system...
Copying backup to the CD or DVD file system...
.......
Building chrp boot image...
Creating Rock Ridge format image: /mkcd/cd_images/cd_image_323664
Running mkisofs ...
....
mkrr_fs was successful.

Making the CD or DVD image bootable...
NFS_Server #
```

This command performs a `mksysb` of the `rootvg` and from that makes a bootable DVD ISO image in */mkcd/cd_images* called *cd_image_323664*. You can either transfer that image to another system (including a PC with a DVD burner) and make a DVD from it, or make a DVD on the local system using the –d *device* argument to `mkcd`. If you want to make CD images, just issue the same command without the –L flag. It then produces multiple CD images if the backup is larger than one CD.

Setting Up NIM

Some people prefer not to make CDs or DVDs and prefer instead to simply leave their `mksysb` images on the file server and read directly from them during a recovery. If you want to do this, you need to use Network Install Manager (NIM). The advantage to this method is that you do not need to load any physical media to perform the restore. The drawback is that your restore is limited by the speed of your Ethernet network, and problems with the Ethernet network can complicate a restore.

If you don't already have a NIM server, you'll need to set one up for this purpose. The good news is that, while NIM is fairly complex, setting up a NIM environment just for this purpose is not that hard. You need to set up the server, add the client to the NIM server, and add a `mksysb` definition for that client.

Setting Up a NIM Server

The purpose of the NIM server is to provide a boot image to the client over the network using BOOTP and TFTP. In our case, we want to boot and install a `mksysb` that we have previously created.

The easiest way to set up NIM is to use the `eznim` wizard introduced in AIX 5.2:

```
NIM_Server # smit eznim
```

For versions before 5.2, there is the regular `smit` panel for NIM, which has a setup option:

```
NIM_Server # smit nim
```

Setting up NIM through the `smit` panels is the easiest and most thorough method. If you choose to install it using the

command-line interface (CLI), read the manpage for `nim_master_setup` for any options to specify relative to your environment. The following instructions create a quick and easy CLI setup:

1. From the latest recommended maintenance level of AIX that you are installing on, find the POWER Version X Volume 1 CD.
2. Insert and mount the CD.
3. Locate and run the `nim_master_setup` command:
4.
```
NIM_Server # nim_master_setup
```
5. This command sets up a default NIM server. It creates */tftpboot* and */export* in the root filesystem.

Adding a Client Definition to NIM

This step adds the host that we want to back up to the NIM server. We need to do this so that the NIM server knows who to serve images to.

The easiest way to add a client is to use the `smit` fastpath:

```
NIM_Server # smit nim_mkmac
```

It can also be installed and set up using CLI. The following instructions apply to a typical client. Both the client and server need to be able to resolve each other via DNS. The example's hostname is *client*, and it has a `chrp mp` architecture that uses twisted pair. For a more detailed usage of the CLI, see the manpage for the `niminit` command. We run the `niminit` command from the client, not the server. This has the added benefit of making sure that the client can talk to the NIM server.

```
client # niminit -a name=client -a master=NIM_server -a pif_name=en0
-a platform=chrp -a netboot_kernel=mp -a cable_type1=tp
```

Setting a mksysb Definition for a Client

Now that we have defined a client, we can select the installation type. This tells the NIM server which image the client should install.

The easiest way to select which `mksysb` to have a client boot is to use the `smit` fastpath. To do this, select `mksysb` as the image resource and set the location:

```
client # smit update_client_eznim
```

In the `smit` panel, select the installation type “mksysb - Install from a mksysb,” and set that to the `mksysb` image that you want to restore.

This can also be set up using CLI. Again, for a more detailed usage of CLI, see the manpage for the `nim` command. In this example, we want the host we are on (*client*) to NIM boot from *NIM_Server* using the `mksysb` file */export/backups/client-6-1-06.msb_image*. We also assign a resource name of `mksysb_res1` to `client-6-1-06.msb_image`:

```
client # nim -o define -t mksysb -a server=NIM_Server –a
location=/export/backups/ client-6-1-06.msb_image mksysb_res1
```

savevg Operations

`savevg` can be used almost identically to `mksysb` except you specify a volume group to back up instead of defaulting to `rootvg`. The volume group must be varied on (that is, active), and the filesystem must be mounted. The exclude list for `savevg` is handled by */etc/exclude.*<*vgname*>, where <*vgname*> is replaced with the name of the volume group that you want to back up.

Using savevg to Back Up a Volume Group

In this example, we back up `datavg` to an NFS share:

```
Client # mount NFS_Server:/NFS_mount/savevg /NFS_mount/savevg
Client # savevg –e –m –f /NFS_mount/savevg/client-datavg-6-1-06.svg_image
datavg
Creating information file for volume group datavg.

Creating list of files to back up.
Backing up 13 files
13 of 13 files (100%)
0512-038 savevg: Backup Completed Successfully.
```

Verifying a mksysb or savevg Backup

Now that you have a backup on tape or disk, you may want to verify the contents of the backup. The most thorough test of a `mksysb` tape is to actually do a restore, but you can get an idea of the integrity of the tape by listing the contents of the archive. If it's a tape, make sure the tape is at the beginning first:

```
# mt -f device rewind
```

Next, use `restore` to list the contents of the tape or file:

```
# restore -s4 -Tvf [no-rewind-device|filename]
```

This shows you all the files that were backed up, the time and date that the backup was run, and how many files were backed up.

Restoring an AIX System with mksysb

This section explains how to completely restore a system using one of the `mksysb` methods discussed earlier.

> **Tip ✎**
>
> To learn how to restore individual files with `restore`, please refer to Chapter 3.

In the event of a disaster, you can boot from tape, CD/DVD, or the network and do a full restore. The processes are very similar, so we'll cover them as one procedure:

1. If the system is still running, you can set it up to boot from tape or CD before you shut it down:
2. a. To boot from CD/DVD:

   b. `Client # bootlist -m normal cd0`

   c. To boot from tape:

   d. `Client # bootlist -m normal rmt0`

   e. To boot from the network:

   f. `Client # bootlist -m normal en0`
3. Power down the system.
4. Attach a system terminal (or monitor and keyboard).
5. Prepare the media:
6. a. If this is a tape restore, attach a tape drive and insert the tape in the drive.

   b. If this is a CD restore, insert the first CD/DVD in the drive.
7. Power up the system.

8. Display the SMS menu. If you weren't able to set the boot list to the proper device before powering down, you need to tell the system to boot from the proper device on power up. On HMC-attached systems, you can select the boot mode before you start the partitions; it then defaults to the SMS menu. On other AIX systems, you need to interrupt the boot process to display the SMS menu. On some older systems, you press F1 while the POST icons are appearing on the monitor. On most later model systems, you press F5 while the POST icons are appearing on the monitor. Consult the hardware guide that shipped with your AIX system for the interrupt sequence for your model of hardware.
9. Once in the SMS menu, select Maintenance Mode, and set one of the following restore/boot options:
10. a. Choose Select Boot Device, and specify CD, DVD, or tape restore.

    b. For a NIM install, select "Installation from Network." Selecting NIM install may require you to supply additional network parameters depending on how your NIM server is set up.
11. The backup is checked for compatibility. If the backup is compatible with the hardware it is being restored onto, it pauses for five seconds and then continues unprompted and restores the `rootvg` based on the settings in *image.data* and/or *bosinst.data*. If you want to interrupt the install, press the 0 key three times (`000`) during the five-second pause. This forces the restore into prompted mode. In the event that the backup does not match the hardware of the system, it automatically begins a prompted restore. The prompted restore looks very much like the initial AIX installer, but at the end you have a restore of your system.

12. If you are not performing a prompted restore, the restore continues uninterrupted until it completes. If you are performing a prompted restore, it allows you to change the physical disk(s) that the restore is performed to, shrink filesystems, etc.
13. When the restore finishes, the system changes its boot device to the install target devices for the `rootvg` and reboot.

System Cloning

Cloning a system is the action of restoring the complete image of one system onto a completely different system. The "restored" system is then a perfect image of the original one. This could be useful in the case of a fire or a similar catastrophe in which the hardware is completely lost. In such a case, you could use an off-site `mksysb` to recover the old system onto new hardware.

AIX 4.x Operating System

The 4.x versions of AIX require special attention because not all the drivers are installed with the BOS. In this case, you are left with two choices:

- Install all device drivers for all available architectures before taking the `mksysb` or backup of the system. That allows the tape to be booted from and restored onto any IBM RS/6000.
- If device drivers are not installed, you can simply boot the new machine from the CD-ROM containing the AIX 4.x operating system. After the boot, choose the `Restore from tape backup` option, insert the `mksysb` tape, and start the restore.

AIX 5.x Operating System

In AIX 5.x, all device drivers and supported kernels are installed with the BOS. This means that a system backup should be largely hardware-independent. However, there are issues when backing up older 32-bit systems. A system backup of a 32-bit AIX system running in 32-bit mode will still run in the same 32-bit mode if installed onto 64-bit

hardware. A backup of a 64-bit system will not run on a 32-bit system because 32-bit hardware cannot run 64-bit software. To change system modes between 32 and 64 bits, see the IBM systems operation guide for the version of AIX you are running.

You should edit the *bosinst.data* file to set the `RECOVER_DEVICES` variable. This specifies whether or not to restore the ODM definitions that were in place at the time of the backup. Since this backup may be restored onto a machine with very different hardware, let AIX rebuild its own ODM definitions at boot.

If you make a bootable CD/DVD on a different platform, you must add the `–G` flag to `mkcd`. This forces it to include the `chrp`, `rs6k`, and `rspc` boot images on the CD.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 14. Mac OS X Bare-Metal Recovery

While Mac OS X is similar to other flavors of Unix in many ways, the topic of backups is one where the differences stand out. Some of the available tools are the same, but familiar tools may not be the best choices. Also, as with other operating systems described in this part of the book, the methods used to record critical metadata and to boot from the recovered disk differ. Following the format of prior chapters, this chapter will show you how to make a simple and inexpensive backup of a Mac OS X system for use in bare-metal recovery.

> **Tip ✎**
>
> This chapter was contributed by Leon Towns-von Stauber. Leon has been using and administering a variety of Unix systems since 1990 and has followed Mac OS X since purchasing a NeXT workstation in 1991.

How It Works

The following sections outline a procedure you can use to anticipate and conduct a full recovery of a Mac OS X system, without needing to install the OS first. The sequence is generally the same as on other platforms.

> **Tip ✎**
>
> A great time to test this procedure is right when you receive your new Mac, and you haven’t yet put any data on it.

You first prepare for the recovery:

1. Attach backup media to the system.
2. Back up the important metadata.
3. Back up the operating system with a native utility.

Then, when bad things happen, you perform the recovery:

1. Boot from alternate media.
2. Partition, and format the new root disk.
3. Restore the operating system information.
4. Set the system to boot from the new root disk.

We'll start with a general overview and then move on to a concrete example of the procedure. As with any backup method, your safety net is only as good as your last backup, so scheduling backups and testing your recovery procedures are important.

Preparing for a Bare-Metal Recovery

This procedure can use any external hard disk drive as a backup medium. Examples include a FireWire or USB storage device of sufficient capacity to hold a full backup. Depending on your storage requirements, even an iPod makes an excellent alternative. Another option is to use a Mac laptop in FireWire Target Disk Mode, assuming it has a FireWire port. Finally, you can also use an NFS share from any Unix-type server. While the USB and FireWire drives may be excellent options for an individual user, the NFS option is usually the best for a data center. Unfortunately, an SMB share will not work, because SMB drivers aren't available from the install CD console that we will be using during the recovery, but more on that later.

> **Tip ✎**
>
> To enter Target Disk Mode, click on Apple Menu→System Preferences→Startup Disk→Target Disk Mode…. Alternatively, hold down the T key while booting the Mac laptop. Hook up a FireWire cable between the laptop and the system you're backing up. The laptop then acts as an external FireWire drive.

After you've attached the external backup disk, you need to collect the information necessary to format the replacement disk before you restore data to it. Save the output of `diskutil list`, `pdisk` *`device`* `–dump` (on PowerPC units), and `mount –t hfs` to files on your backup disk that you can reference during recovery.

> **Tip ✎**
>
> As of the time of this writing, Apple has recently released the first Macs built around Intel x86 CPUs instead of PowerPC CPUs. If you happen to be running an Intel-based Mac, you will not be able to use `pdisk`, and `fdisk` will not supply information on the disk you're running it from, but more on that later.

It's also a good idea to save Open Firmware variable settings to a file with `nvram –p`, just in case your NVRAM gets zapped along with your root disk. And for good measure, keeping a copy of the output from System Profiler handy means you have fairly complete documentation of your system's hardware and software setup available. (Launch the System Profiler application by going to the Apple Menu and selecting About This Mac→More Info…,

then use File→Save to create a profile of your configuration.)

Now it's time to back up the data you'll use to recover the operating system. There are several native backup utilities from which to choose, but for this procedure, we use one specific to Mac OS X named `ditto`. With Mac OS X 10.4 and later, you could conceivably choose `tar` instead of `ditto`. However, we're going to use an install disc as the boot device during recovery, and the compression tools employed by `tar` (`gzip` or `compress`) aren't available there, whereas `ditto` features have built-in PKZIP compression. In order to save space on the backup medium and for compatibility with older versions of the OS, the example procedure detailed later in this chapter uses `ditto` for backup and recovery. For more on backup utilities native to Mac OS X, see Chapter 3.

Performing a Bare-Metal Recovery

When the day comes to use your backup for a system recovery, either as practice or because something terrible has happened, your first task after replacing any failed components is to boot up the system. The easiest choice for this is to use the same optical media you would use to install a fresh copy of Mac OS X, and that's the approach we take in our example.

Once the machine has booted up, it's time to partition and format the new disk using the `diskutil` and `pdisk` output you saved during backup. For this you can use the `diskutil partitionDisk` command.

Restoring the data is pretty much the opposite of backing it up: use `ditto` to decompress and unpack the archive files. If you need to restore your Open Firmware setup, use `nvram -f` *`filename`*. Finally, once all the data has been

restored, use the `bless` command to configure the system to boot from the new disk.

A Sample Bare-Metal Recovery

This example uses `diskutil`, `pdisk`, `mount`, `nvram`, `ditto`, and `bless` to back up and recover a complete Mac OS X system on an Apple iBook, using an iPod as the backup medium. Some alternate examples will also be provided throughout the sample procedure. All volumes are formatted as HFS+.

Perform the Backup

First, attach the backup disk to the system. The disk volume is automatically mounted by `diskarbitrationd`; for this example, we assume it's mounted under */Volumes/iPod*.

> **Tip ✎**
>
> The target disk could also be a share mounted from an NFS server. To do that, you need to mount it. For example, if there is an NFS share called */backups* on the NFS server *192.168.0.1*, you can mount it to the directory */backups* by running the following commands:
>
> ```
> # mkdir /backups#mount -t nfs 192.168.0.1:/backups /backups
> ```

Now save the important metadata. Use `diskutil` to get partition names and sizes, as well as the root disk device name (usually */dev/disk0*). With this in hand, run `pdisk` to get exact partition sizes in blocks, and `mount` to take note of the root partition and any special partitioning options (such as journaling or case-sensitivity), as shown in the following example:

```
% mkdir /Volumes/iPod/Backup
% diskutil list | tee /Volumes/iPod/Backup/diskutil.txt
/dev/disk0
   #:                       type name                 size
identifier
   0: Apple_partition_scheme                        *18.6 GB  disk0
   1:    Apple_partition_map                        31.5 KB   disk0s1
   2:              Apple_HFS Mac OS X               7.9 GB    disk0s3
   3:              Apple_HFS Mac OS X Alt           7.9 GB    disk0s5
   4:              Apple_HFS Local                  2.5 GB    disk0s7
/dev/disk1
   #:                       type name                 size
identifier
   0: Apple_partition_scheme                        *18.6 GB  disk1
   1:    Apple_partition_map                        31.0 KB   disk1s1
   2:             Apple_MDFW                        32.0 MB   disk1s2
   3:              Apple_HFS iPod                   18.6 GB   disk1s3

% sudo pdisk /dev/disk0 -dump | tee /Volumes/iPod/Backup/pdisk.txt

Partition map (with 512 byte blocks) on '/dev/disk0'
 #:                type name                   length   base
( size )
 1: Apple_partition_map Apple                       63 @ 1
 2:          Apple_Free                        262144 @ 64
(128.0M)
 3:           Apple_HFS Apple_HFS_Untitled_1 16515072 @ 262208
(  7.9G)
 4:          Apple_Free                        262144 @ 16777280
(128.0M)
 5:           Apple_HFS Apple_HFS_Untitled_2 16515072 @ 17039424
(  7.9G)
 6:          Apple_Free                        262144 @ 33554496
(128.0M)
 7:           Apple_HFS Apple_HFS_Untitled_3  5253424 @ 33816640
(  2.5G)
 8:          Apple_Free                            16 @ 39070064

Device block size=512, Number of Blocks=39070080 (18.6G)
DeviceType=0x0, DeviceId=0x0

% mount -t hfs | tee /Volumes/iPod/Backup/mount.txt
/dev/disk0s3 on / (local, journaled)
/dev/disk0s5 on /Volumes/Mac OS X Alt (local, journaled)
/dev/disk0s7 on /Volumes/Local (local, journaled)
/dev/disk1s3 on /Volumes/iPod (local, nodev, nosuid)
```

Tip ✎

Again, if you're on an Intel Mac, `pdisk` isn't available, and `fdisk` will not report on the disk it's running from, so you need to do something else. One choice is to use the `diskutil` output, although it won't be as exact. Another choice is to boot from the install media and launch a Terminal, then run `fdisk` from there, storing the results in a file on your backup drive.

Now save your Open Firmware variables with the `nvram` command:

```
% sudo nvram -p > /Volumes/iPod/Backup/nvram.txt
```

As mentioned earlier, at this point you can also save information from System Profiler.

Now it's time to back up the data. With the system in a quiescent state to avoid filesystem inconsistency, back up each partition on the root disk:

```
% cd "/Volumes/Mac OS X"
% sudo ditto -c -k -rsrc -X . "/Volumes/iPod/Backup/Mac OS X.zip"
% cd "/Volumes/Mac OS X Alt"
% sudo ditto -c -k -rsrc -X . "/Volumes/iPod/Backup/Mac OS X Alt.zip"
% cd /Volumes/Local
%sudo ditto -c -k -rsrc -X . /Volumes/iPod/Backup/Local.zip
```

These commands create a ZIP archive for each partition you're backing up. All resource forks and file metadata are retained in the archive files.

Tip ✎

If you really prefer to use `tar` with Mac OS X 10.4 or later, replace the `ditto` commands with something like this:

```
% sudo tar clpzf /Volumes/iPod/Backup/partition.tgz .
```

Also, make a copy of */usr/bin/gzip* to the backup drive. During recovery, add the directory containing the copy of `gzip` to your command path:

```
# export PATH=${PATH}:/Volumes/iPod/Backup
```

Now, restore the data like this:

```
# cd /Volumes/partition#tar xpzf /Volumes/iPod/Backup/partition.tgz
```

Detach the backup disk and keep it somewhere safe. (If you're using NFS, you only need to unmount it.)

Recover the System

You have a Mac OS X system with a nonfunctional root disk. If you followed the procedure covered earlier, the recovery should be straightforward.

First, turn on power to the system. While the system is starting up, hold down the `Option` key until the machine presents a graphical list of devices from which to boot. Insert a Mac OS X installation CD or DVD (disc 1 if it's a multidisc set), and click on the `reload` button to make it show up in the list of boot devices. Select the installation disc, and proceed with the boot. After booting up and

selecting the language to use, you reach a welcome message, and the menu bar shows up at the top of the screen. From the `Utilities` menu, launch `Terminal` to get a command line.

Now attach the backup disk. As with the backup procedure, `diskarbitrationd` handles the mounting. When this is done, you have the following storage devices attached: an optical drive with a Mac OS X install disc mounted on */*, an external backup disk mounted on */Volumes/iPod*, and a new internal hard disk with nothing mounted. (If for some reason there are partitions mounted from the internal disk, unmount them at this point with `diskutil unmountDisk disk0`.)

Tip ✎

An alternative to using a USB or FireWire drive, such as an iPod, is to use an NFS mount. Besides an NFS server, you will need an IP address. If you have a DHCP server available, you are given an IP address automatically. If you don't have a DHCP server, you have to manually set the IP address using the following command:

```
# ifconfig en0 ipaddress netmask netmask broadcast broadcast-address
```

You then need to mount the NFS share. Assuming an NFS server with an IP address of *192.168.0.1*, and a share called */backups*, the command to mount it is:

```
# mkdir /var/tmp/backups#mount -t nfs 192.168.0.1:/backups /var/tmp/backups
```

You then will be able to access the backups on the */backups* file share. Remember to unmount the share after you've restored data from it.

Refer to */Volumes/iPod/Backup/diskutil.txt* and */Volumes/iPod/Backup/pdisk.txt* so that you can partition the new disk. The `diskutil` output provides the partition names. To determine the size of each partition, use the `pdisk` output to add up the partition's block size with that of the free area listed before it, then divide by two to convert 512-byte blocks to kilobytes. For example, the sizes of the first two partitions in this example are (16,515,072 + 262,144) / 2 = 8,388,608, and the size of the last partition is (5,253,424 + 262,144) / 2 = 2,757,784.

> **Tip ✎**
>
> We go through this calculation because specifying the sizes in gigabytes as shown in the `diskutil` output usually results in partitions sized slightly larger or smaller than you'd expect. If you're recovering an Intel Mac, though, the `diskutil` output may be the only information you have.

Use `diskutil` to partition the disk:

```
# diskutil partitionDisk disk0 3 JournaledHFS+ "Mac OS X" 8388608K \
JournaledHFS+ "Mac OS X Alt" 8388608K JournaledHFS+ Local 2757784K
Started partitioning on disk disk0
Creating Partition Map                                    5% ..
Formatting Disk                                          32% ..
Formatting Disk                                          54% ..
Formatting Disk                                          100% ..
Finished partitioning on disk disk0
/dev/disk0
   #:                       type name            size
identifier
   0: Apple_partition_scheme                     *18.6 GB  disk0
   1:    Apple_partition_map                     31.5 KB   disk0s1
   2:              Apple_HFS Mac OS X           7.9 GB    disk0s3
   3:              Apple_HFS Mac OS X Alt       7.9 GB    disk0s5
   4:              Apple_HFS Local              2.5 GB    disk0s7
```

See the `diskutil` manpage for more details on the command syntax and options. One thing to note is that if you'll be running Classic for Mac OS 9 support on this system, you should add the `OS9Drivers` parameter after the argument specifying the number of partitions. You can tell if your old disk had OS 9 drivers installed by the presence of several additional `Apple_Driver` partitions in *diskutil.txt*.

After you partition the disk, the new partitions are mounted automatically. Now you're ready to restore the filesystem data. If need be, you can also reset the Open Firmware variables:

```
# ditto –x –k –rsrc "/Volumes/iPod/Backup/Mac OS X.zip" "/Volumes/Mac OS X"
# ditto –x –k –rsrc "/Volumes/iPod/Backup/Mac OS X Alt.zip" "/Volumes/Mac OS X
Alt"
# ditto –x –k –rsrc /Volumes/iPod/Backup/Local.zip /Volumes/Local
#nvram –f /Volumes/iPod/Backup/nvram.txt
```

The last step of recovery is to prepare the new disk for booting. The following `bless` command enters the designated folder's directory ID (an HFS+ identifier analogous to a UFS inode number) into the disk's Master Directory Block, and sets the `boot-device` Open Firmware variable so that the system boots from the new disk the next time it comes up:

```
# bless -folder "/Volumes/Mac OS X/System/Library/CoreServices" -setBoot
```

Tip ✎

Upon booting, the system will look in the *blessed* folder for a file of type `tbxi` and find *BootX*, which contains bootstrapping code to begin the kernel initialization process. You can use the following to see this file's type on your system if you have the Developer Tools installed:

```
% /Developer/Tools/GetFileInfo \/System/Library/CoreServices/BootX
```

Again, if you need Mac OS 9 support on this system, you need to add something to the command line, like this:

```
# bless -folder "/Volumes/Mac OS X/System/Library/CoreServices" -setBoot \
-folder9 "/Volumes/Mac OS 9/System Folder" \-bootBlockFile
/usr/share/misc/bootblockdata
```

See the `bless` manpage for more information.

Finally, reboot the system and get back to work (or play, as the case may be).

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Part 5. Database Backup

Part V consists of eight chapters:

Chapter 15, Backing Up Databases

Provides an overview of database backup, concepts, terminology, and procedures.

Chapter 16, Oracle Backup and Recovery

Explains how to perform Oracle hot backups using `rman` or user-managed backups.

Chapter 17, Sybase Backup and Recovery

Shows exactly how to use the Backup Server to back up Sybase ASE.

Chapter 18, IBM DB2 Backup and Recovery

Explains how to perform backup and recovery of DB2 databases.

Chapter 19, SQL Server

Explains how to back up and recover SQL Server databases.

Chapter 20, Exchange

Explains how to perform backup and recovery of Exchange databases using the built-in `ntbackup` plug-in for Exchange.

Chapter 21, PostgreSQL

Explains backup and recovery of PostgreSQL databases.

Chapter 22, MySQL

Provides an overview of the various backup and recovery options available for MySQL.

Chapter 15. Backing Up Databases

Performing regular database backups is one of the hardest tasks that lies before today's system administrator. The primary reason is that databases are infinitely larger and more complex than simple filesystem files. In order to properly back up a database, you first need to:

Understand the internal structure of your database

Understand the available utilities

Have an excellent working relationship between system administrators, storage administrators, and database administrators

Once you've accomplished all of that, you need to choose among your various options:

Shut down the database and back up its files "cold."

Put the database in a mode that allows its files to be backed up live, or "hot."

Use its built-in tool to back up to disk or tape without a commercial utility.

Buy an agent for a commercial utility that will back it up to its devices.

Almost anyone who reads this list will find at least one of these steps daunting. Many people work with databases that operate 24 hours a day, 7 days a week. They can't shut them down for hours at a time to back them up. Even if they could, if a database uses raw devices, it can't be backed up with a filesystem backup utility. Of course, `dd` would work on Unix, but that would mean doing one thing for filesystems and a different thing for databases. A common theme throughout this book is that different is bad. Every special case is a chance for failure. It's

something else you have to code for, something else you have to watch—something else to break. The result is that database backups are not easy.

Part of the problem is the design process of the actual database engine itself. Historically, the need for bigger storage and faster queries drove the design of a product to develop much more than its ability to back itself up. Over the last few years, databases have grown from a gigabyte or so to well beyond several terabytes. This growth in size and performance happened because the customer base screamed for it. Unfortunately, they weren't simultaneously screaming for a backup utility to support those huge databases.

Can It Be Done?

Think about database backups from a big-picture perspective, and compare them to filesystem backups. There are a number of good backup utilities on the market now. Why aren't there just as many for database backups? There is certainly a demand for them.

One of the reasons is the complexity of the task. In order to release a database backup product, a company needs to consider several factors:

Multiple moving targets

> How do you get a database to hold still? Have you ever tried to take a picture of 100 people? Designing a backup utility for a database is very hard because you have to "take a picture" of hundreds of files all at once.

Interrelationship between the files

> A database backup program needs to understand all the database elements and how they relate to each other.

Necessity to work with the database engine

: This is essential. If the database understands that you are running a backup, it can help you. If you don't interface with the database, you'll be backing up blind.

The size of the job

: How do you get more than a terabyte of data to a backup drive in one hour? That's what some of the backup requirements are!

Recoverability versus cost

: You need all of the preceding, but you don't want to mortgage your house to get them.

Differing levels of automation

: Different customers want different levels of automation. Some want everything managed by the library while others would rather do it themselves.

With these kinds of requirements, is there any hope of getting utilities that are up to the challenge? The answer is "Yes!" Database companies have finally recognized that backup utilities do have an effect on the overall sales of a database engine. They have finally started producing good utilities that interface with other commercial backup products. Vendors have even taken an active lead in making sure that customers use a utility that works properly on both the backup side and the restore side. Now that there are decent backup utilities, though, you have another problem—confusion.

Confusion: The Mysteries of Database Architecture

Any system administrator who has been in the business for any length of time can probably tell you how to back up the home directories on any system. Start asking about backing up databases, though, and even the most seasoned veterans may start to squirm. The architecture of a database is a mystery to many administrators. Unfortunately, you really need to understand how the database works to properly recover from a disaster. They know how to back up a filesystem, but ask them to find the backups for data space A in database B in instance C, and they look at you with fear in their eyes! They just don't have any experience in database design, nor do they have time to get that experience. If they work in a heterogeneous shop with more than one database, it gets even harder. Their only hope is that the database administrators know what they are doing.

DBAs know all about database architecture. They also know how to back up their database to disk or maybe to a standalone backup drive. They might not have any experience with commercial backup software or large, automated libraries. If the database is too big to back up to disk, and they don't have a standalone backup drive to back up to, they will have to work with the system administrators (SAs) to get the backup done. The only problem is they don't share a common language with the SAs because the SAs don't understand database architecture.

Database products also differ from one another. Try to get an Informix DBA and an Oracle DBA to agree on what a tablespace is! One of the reasons this is difficult is that

different products use the same term for different logical elements. What Informix calls a tablespace, Oracle calls a segment. Don’t confuse that, however, with a Sybase segment, which is closer to what Oracle calls a tablespace. Are you confused yet? Don’t worry.

The Muck Stops Here: Databases in Plain English

This chapter explains all the basics you need to know in order to learn more about database backup and recovery. You then will be prepared to read and understand the vendor-specific database chapters in this book as well as any of the appropriate sections of your favorite DBA book or manual. This chapter assumes no prior knowledge, beginning with some of the very basic elements such as tables and rows, so that even the most junior person can understand everything. However, that does not mean that a seasoned SA or DBA should not read this chapter. It took several experienced DBAs of a number of different types of databases to come up with the information you see here, so almost any DBA should be able to learn something from this chapter.

This chapter includes useful tables that list all of the elements of database storage for each of several databases: DB2, Informix,[1] MySQL, Oracle, PostgreSQL, SQL Server, and Sybase. If you're not a DBA, the information here will allow you to discuss backup matters intelligently with your DBA. If you are a DBA for one product, it will help you to discuss backup and storage strategies with DBAs for other products—without getting confused! The desired result for your organization is that you can all agree on why and how to protect your database data. What a beautiful world that will be!

---

[1] *Unix Backup & Recovery* had an Informix backup chapter. It did not make it into this book for space reasons, but it is still available online at http://www.backupcentral.com.

What's the Big Deal?

Why are database backups so hard? Can't I just shut down the database and back up the whole system? These are all questions that may be going through your head. If you already know the answers, feel free to skip to the next section.

Why are we hearing so much about database backups?

> The demand for relational database management systems (RDBMSs) has grown exponentially in the last few years. Not only are there more databases, they are faster, larger, and more complex than ever before. Companies are relying increasingly on bigger and bigger databases to store their information—information that, if lost, would be irreplaceable. Customers have started to recognize the importance of safeguarding their data, and the demand for better backup and recovery utilities has followed. Database companies and backup product companies have finally responded with utilities that are up to the task at hand.

Why is backing up a database so hard?

> Actually, backing up a database isn't that hard. It's restoring the database that has caused many people to go insane! Seriously, though, the reason it is so difficult is that you need a good SA and a good DBA to design a good backup plan. Most people know only one side well. If you or your people know both sides, then consider yourself very lucky. In some companies, it's tough to get both sides to work together.

Can't I just shut down the database and back up the whole system?

In some cases, yes. There are a number of considerations that might prevent you from doing this, including your database platform, whether you are using raw or filesystem files, and whether you need point-in-time recovery—not to mention the loss of availability of the database during such an operation. Those details are covered in appropriate chapters in this book.

Do I have to buy a commercial utility to back up my database?

If you can either back up your database to disk or back its files up hot or cold using a standard backup utility, then the answer is no. If you can't do these things, then you're probably going to need a commercial utility.

Database Structure

There are many terms thrown around when discussing RDBMSs. The good news is that you don't need to know all of them to properly back up and recover databases. You do need to know some of them, though—about 20 individual terms. It's helpful to know:

- What all the different storage elements are and what they are called
- How these elements are logically organized within the RDBMS
- What facilities are in place to protect and back up the data

This information can be complex because it depends a lot on how you look at the data. This chapter presents this information from first a power user's, then a DBA's, point of view. The various building blocks of a database are defined, although we may have to go up and down the building a bit before we're done!

> **What About Exchange?**
>
> Chapter 20 covers Exchange, but Exchange is kind of the "odd man out" here. While Exchange is at its heart actually a relational database, Microsoft does a pretty good job of hiding that from its administrators and users. They don't use any of the usual database terminology when talking about Exchange. This chapter covers Exchange where it can, but Exchange is not covered as much as the other products.

The Power User's View: Logical Elements of a Database

Before looking at how databases are stored on disk, let's look at the "power user's" view of a database. This is necessary because some of these terms are used in the definition of the storage elements. We're calling it the "power user's" view because many users will have little or no knowledge of any of these terms. But unless they want to start doing the DBA's job of putting a database together, these terms may be all that they will ever need. The terms are presented in no particular order because it is very difficult to define one term without using another one. Therefore, it may help some readers to study this section more than once.

This view also could be called the "logical" view, because many of the elements described in this view don't exist in a physical sense. That fact is one of the many things that differentiate an RDBMS from a simple spreadsheet. A spreadsheet has one table that resides in one physical file. An RDBMS table, on the other hand, gives the appearance that its data is all sitting in one place, but it may be spread out all over the system.

Instance

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Instance	Server	Instance	Instance	Instance	Cluster	Instance	Server

Instance is probably the most difficult term to explain because it means different things to different people—and to different database platforms. The simplest definition is that an instance is one or more processes on one or more machines, through which the databases on that machine (or set machines) communicate with shared memory. There can be multiple databases within an instance, and a database also can be distributed across multiple instances on the same machine or on separate machines within a cluster. Therefore, an instance and a database are two entirely different concepts.

The Sybase term *server* (or *dataserver*) stems from the original intent that each *machine/server* would have one Sybase *instance/server* on it, although it is now quite common to have more than one instance on each machine. Informix occasionally uses the term in this manner, and it can be rather confusing. They tend to use *server* when speaking about the software and *instance* when speaking about a running environment, especially when discussing running multiple instantiations of the server.

If an instance needs to be shut down and restarted for any reason, all databases within that instance are unavailable during the shutdown. This may help you understand the original definition, because all the databases within an instance have a single connection to shared memory, which is provided by the instance. If the instance is shut down, that connection is no longer available. (See Figure 15-1 for a graphical representation of an instance.)

Database

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Database	Database	Database	Database	Database	Database	Database	Database

A *database* is a collection of database objects. It may be a very simple database with one table and no indexes, or it could contain many tables, indexes, and other database objects. (All database products can have more than one database object and more than one type of database object.) For example, the “customer” database may contain a table that has customer addresses and an index for that table. It also may contain a binary large object (BLOB) table that contains a scanned-in image of the customer’s contract, a regular table that contains the data from that contract, and an index for that table. Then there might be another database that keeps track of all the widgets that your company sells. (See Figure 15-1 for a graphical representation of a database.)

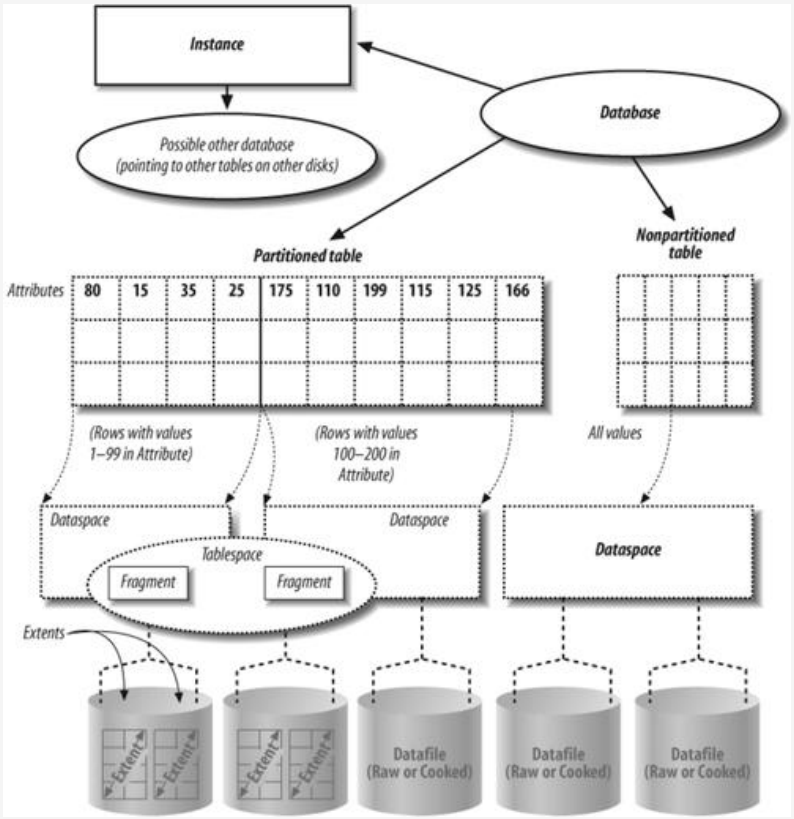


Figure 15-1. A database instance

Table

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Table	N/A	Table	Table	Table	Table	Table	Table

A *table* is a grouping of related information. (That is why it is called a *relation* in formal database terminology.) Information is typically grouped in such a way that data is not replicated between tables except when necessary. In the previous example, the customer database could contain the customer information table, with each customer having a unique account number. The BLOB table that stores the customer's signed contract would then need to store only the customer's account number to be able to tie the two pieces of information together. (BLOB data is discussed later in this chapter.) That method takes up less space than storing the customer's whole name with the contract. The order table would contain that account number as well, and it might list what a customer ordered only by part number. If you wanted to see the details about that part number, the instance could reference the "parts" database using that part number. (See Figures 15-2 and 15-3 for graphical representations of a table.)

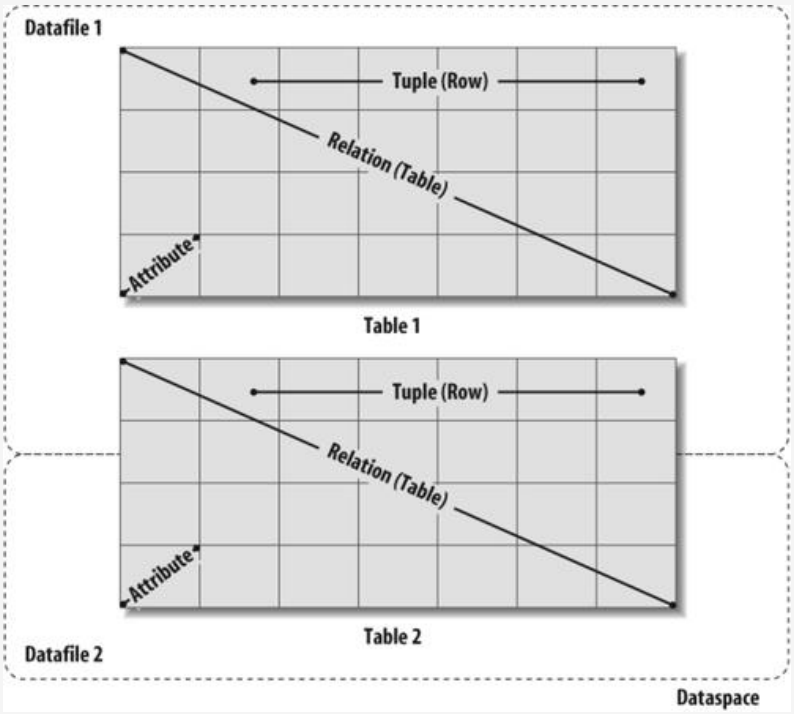


Figure 15-2. Table layout

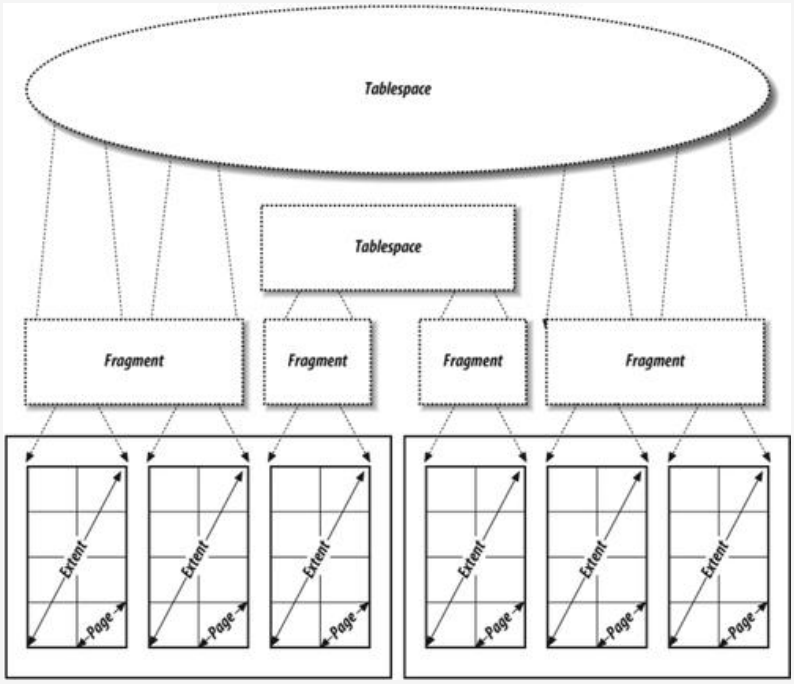


Figure 15-3. Tablespace layout

A related term is a *view*, which usually refers to a *virtual* table. For example, you may put together a view that references the customer, order, and parts tables to present a united view of information about a given customer. Data often is replicated in multiple views, but because it’s a virtual table, it

doesn’t take up extra storage space on the disk. A view normally is constructed on the fly from the results of a `SELECT` statement.

Index

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Index	N/A	Index	Index	Index	Index	Index	Index

An *index* is a special-purpose object that allows for quicker lookups of a normal table. A table is indexed by the value that you normally would use to look up a record (row). For example, the customer database might be indexed by last name if customers frequently call in and do not know their account number. It has a unique ability when recovering a database, because you can usually recreate an index from an existing table instead of recovering it.

LOBs

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
BLOB, CLOB, DBCLOB	Email	BLOBspace	*blob and *text datatypes	BLOB, CLOB, BFILE	ByteA, BLOB, and text	Ntext, text, and image datatypes	Character or image datatype

Large object (LOB) data has grown in popularity within the last few years. It refers to anything that does not fit into a “normal” table. This may range from a large piece of text data to a scanned-in image. Most databases differentiate between character LOB (text) and binary LOB (graphics and the like) data.

If the LOB data is stored inside the database, it presents no unique backup requirements. However, the use of datatypes that allow the storage of the LOB data outside the database (i.e., in the filesystem) are a different story. While they may provide many performance enhancements, they do present a unique backup challenge. The BLOB data needs to be backed up in sync with the database data because the database is keeping track of what files are where. Suppose a piece of BLOB data was inserted at 11:00. If you back up the filesystem at 10:00 and the database at 12:00, the database will know about a file that exists out on the filesystem, but that file will not be found on your filesystem backup. Even more confusion may be added if the filesystem backup spans the time of the database backup. In other words, it begins at 10:00 and ends at 4:00, although the database backup begins at 12:00 and is done by 2:00. (PostgreSQL does allow you to store LOB data in the filesystem, but it will be backed up by default in later versions of PostgreSQL.)

There are only two ways to resolve this conflict. The easiest way is to shut down the database or put it in read-only mode during the entire time of your filesystem backup. This may be impractical for many environments.

The second way is to use the *snapshot* concept, which allows you to take a "snapshot" of the entire filesystem in just a few seconds and then take all night to back it up. This provides a consistent picture of the filesystem at a certain point in time.

Microsoft does not really talk about emails being LOBs. But if you think about it, that's what they are. An email can be anything from a simple line of text to a huge attachment, and all of these are stored in the Exchange database. Therefore, they must be treating these as LOBs from a database perspective.

Object

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Object	N/A	Object	Object	Object	Object	Object	Object

This generic term refers to any element managed by a database, and there are several object types, including tables, indexes, stored procedures, functions, synonyms, and triggers. So we use the term *object* to describe any type of element that may be in a database. This includes, but is not limited to, simple tables, indexes, BLOB tables, stored procedures, packages, and triggers.

Row

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Tuple	N/A	Tuple	Tuple	Tuple	Tuple	Row	Tuple

A row (called a *tuple* in formal database terminology) is a collection of related attributes. For example, there may be a row that contains all the basic information about a customer, such as her name, address, account number, and phone number (this is also similar to a row in a spreadsheet). Each row typically has at least one unique attribute, such as the account number, to distinguish it from other rows. A row is also sometimes called a *record*. (See Figure 15-2 for a graphical representation of a row.)

Attribute

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Attribute	N/A	Attribute	Attribute	Attribute	Attribute	Attribute	Attribute

An *attribute* is the basic element of data within a table. (See Figure 15-2 for a graphical representation of an attribute.) It is a single value, such as a customer's name or zip code. An attribute may be very small, such as a zip code, or very large, such as a BLOB. An attribute is the value that a database user changes when performing a transaction. Transactions are covered later in this chapter.

The DBA’s View: Physical Elements of a Database Environment

The DBA has to know quite a bit more about the database than even the most sophisticated power user, because the DBA must create databases, tune them, back them up, and recover them in the case of failure. DBAs also know some variation of a programming language called SQL that allows them to construct precise types of queries that increase the usability of the database. Good DBAs also need to know quite a bit about operating system technology in order to efficiently use the storage capabilities of their operating environment.

The good news is that, unless you’re a DBA, you don’t need to know all of that to properly back up and recover databases. This section describes the physical elements of database storage and how they are combined with the logical elements discussed earlier.

The bad news is that, unlike the terms in the user’s view, the elements in the database view are called something different in almost every product. Often, the same term presented in the previous section is used to describe different types of elements in different products. It took quite a bit of work to be able to discuss them all in a single chapter. It often was very difficult to find a generic term that would apply to all of them without confusing things. Therefore, the terms that serve as headings for these sections are often words that were coined just for this purpose. This generic term is used throughout this chapter when discussing the different types of storage elements and how they fit together. The product-specific terms are used only when referring to the products themselves.

> **Tip ✎**
>
> Some of the coined terms that follow may be useful only when discussing things with someone else who has read this chapter. If you are discussing storage elements with a particular DBA who has not read this chapter, be sure to use the appropriate terms for the product that DBA knows.

Page

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Page	Page	Page	Page	Block	Page	Page	Page

The *page*, also called a *block*, is the basic building block of every database. It is the smallest amount of data that is moved in an I/O operation. It is similar to, although not exactly the same as, a filesystem block. (You can have a 2 K

block inside a database that sits on a filesystem with an 8 K block size.[2]) Page sizes tend to range from 2 K to 32 K in size, but some database products allow you to specify a custom page size for your environment. Whatever size the page is, it is the smallest atomic entity within a database. When you modify a table within a database, it eventually modifies one or more pages stored somewhere on disk. (See Figures 15-2 and 15-3 for graphical representations of a page.)

Datafile

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Container	Datafile	Chunk	Datafile	Datafile	Datafile	Datafile	Device

A *datafile* is where the data is stored. This may be a *raw device* (e.g., */dev/hda1* in Linux), or a *cooked file* (e.g., */oracle/data/dbs01.dbf* or *c:\database\somefile.dbf*). Some products require the use of raw partitions, while others merely suggest it. Some products allow a mixture of raw and cooked files. The only real difference to the DBA is how they are initially created and how they are backed up. Other than that, they look the same within the database. (See Figure 15-3 for graphical representations of a page.)

Extents

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Extent	N/A	Extent	Extent	Extent	Extent	Extent	Extent

An *extent* is a number of pages that are logically grouped together and are considered *logically contiguous*. They may or may not be physically contiguous. (If the pages in an extent are physically contiguous, that means that they are physically next to each other.) Informix extents are physically contiguous, while others may or may not be, depending on when and how they were created. All extents are considered logically contiguous, because they are treated as a single block of storage that is allocated to a table. Extents do not span more than one datafile, but a datafile may contain any number of extents. The actual size of an extent is determined by the database platform. (See Figures 15-1 and 15-3 for graphical representations of an extent.)

Tip ✎

Informix DBAs: the term *tablespace* in this chapter refers to an Informix *dbspace*. A *tablespace* as defined here is "the space into which you insert tables." Informix uses the term *tablespace* (and a similar word *tblspace*) to mean something different. As you can see in Figure 15-1, a tablespace can consist of extents that are in different datafiles. When a table spans datafiles like this, Informix uses the term *tblspace* to refer to the part of a table that resides within a single datafile, and the term *tablespace* to refer to the amount of space a table takes up.

Tablespace

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Tablespace	Store or storage group	Dbspace	Tablespace (InnoDB and NDB)	Tablespace	Tablespace	Filegroup	Segment

A *tablespace* is a collection of one or more datafiles and is the space into which you insert tables. (See Figure 15-3 for a graphical depiction of a tablespace.) A table is created in a tablespace (e.g., `create table in tablespace alpha`). When creating an instance in most database products, you specify which datafile will be the main (or system) tablespace. The database product then creates this tablespace for you automatically.

All Sybase databases have at least two *segments*: *default* and *system*. They're automatically created when you run the `create database` command. You can also create your own segments that consist of one or more Sybase devices and can specify those segments in the `create table` command. System tables reside in the system segment. If you don't create your own custom segments, all user tables will be placed in the default segment. This means that segments are the same as tablespaces; however, since many Sybase DBAs do not define custom segments, they do not think of segments as tablespaces. SQL Server calls its tablespaces *filegroups*: the primary filegroup stores system tables, and the default filegroup is the default location for tables. You can also create your own filegroups and create tables in them as well. Since a given user's data in Exchange is stored within a single storage group, and there is a table of users, we say that a *store* or *storage group* is equivalent to a tablespace. In MySQL, only tables stored in the InnoDB and NDB storage engines have tablespaces.

Partition

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Partition	N/A	Fragment	Partition	Partition	Partition	Partition	Partition

One of the biggest advancements in RDBMS technology is the ability to spread, or *partition*, a table across multiple resources. Historically, a table had to be contained within a tablespace, as described earlier. Now some database products allow you to specify that a table is partitioned across multiple tablespaces based on the values of certain attributes. For example, you could create a table that is partitioned across tablespaces A and B. You could specify that all records with value 1–100 in attribute A go to tablespace A, and all records with value 101–200 in attribute A go to tablespace B. (See Figure 15-1 for a graphical depiction of a partitioned table.) A DB2 partition allows you to allocate a table to multiple CPUs. A partitioned table does not present any unique backup requirements.

Master database

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Catalog	Private and public information store databases	Sysmaster, *onconfig* file, *rootdb*	MySQL database	Control file	System tables	Master database	Master database

Each instance has some way to keep track of all the storage elements it has at its disposal. This *master database* keeps track of all the devices and their status, and any information that all the databases need to have access to. If multiple databases are allowed, it needs to track them as well. Sybase has a special database to do this, and Oracle has what it calls a *control file*, which keeps track of this information. Informix also has a special database, called the *Sysmaster*, that tracks the status of every object within an instance. However, some information is tracked by the *onconfig* file and reserved pages within the *rootdb*.

Transaction

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Transaction	Transaction	Transaction	Transaction	Transaction	Transaction	Transaction	Transaction

A *transaction* is an activity within a database that changes one or more attributes within one or more tables. If a user changes a customer's address, that is a transaction. There are two types of transactions, a simple transaction and a complex transaction. A *simple* transaction is done in one statement (e.g., `update attribute X in table Y to 100`). A *complex* transaction may be much longer, and opens with a *begin transaction* statement and closes with an *end transaction* statement. There may be a number of simple statements in between the open and close statements, or there may be a complicated SQL program that updates hundreds of values based on certain parameters. For example, it has become relatively common to change someone's area code as a city grows and splits into multiple area codes. A complex transaction could be designed that would scan all

customer phone numbers and change their area code based on their three-digit exchange. A complex transaction is treated as an “atomic” event—it’s all or nothing. Both the start transaction and end transaction statements are recorded in the transaction log (defined later), and if anything happens before the end transaction statement is recorded, all changes that were made by that transaction are *rolled back*, or undone. Things that can make that happen include the user logging out in the middle, the database being shut down, the system crashing, or even the user changing his mind.

Not all storage engines in MySQL support transactions. In those ACID-compliant tables that support them, they would be called transactions. However, the default storage engine (MyISAM) does not support transactions.

Rollback log

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Transaction log	Transaction log	Physical log	Rollback segment (InnoDB)	Undo segment, rollback segment	Internal table	Transaction log	Transaction log

From a data integrity standpoint, it is important to realize what a transaction does. While to a user’s eyes a transaction changes a record in the database, it actually changes one or many pages. It is on the page level that transaction recovery is done. If a given transaction modifies 100 pages, and the transaction does not complete, those 100 pages must be returned to what they looked like before the transaction occurred. This is referred to as *rolling back* the page to its *before image*. (The before image is what the page looked like *before* it was changed.) The following elements describe the facilities that databases have to ensure that this (and other data-integrity activities) occurs properly.

The *rollback log* is the place where the database stores this before image. Informix and Oracle have a dedicated log just for this purpose. The before image of each changed page is stored in this log until the transaction is complete, or *committed*. Sybase, SQL Server, and DB2 record both the before image and the transaction data in the *transaction log*. It is important to note that this before image must be physically written to disk before any pages are to be physically changed. That ensures that the before image is available if the system crashes. How this before image is actually used varies widely between database products. Oracle has changed how rollback works, and now uses the undo segment instead of the rollback segment. PostgreSQL stores the previous records for changed rows in the table itself. At some point, previous records are deleted via a vacuum process.

Transaction log

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Transaction log	Transaction log	Logical log	Transaction log	Redo log	Write ahead log	Transaction log	Transaction log

Suppose that a system were to crash in such a way that it needed to be recovered from your latest database backup. If there were no way to redo the transactions that occurred since the last backup, all such transactions would be lost. A *transaction log* records each transaction and what pages it changed. This information is used in case a system crash requires reentering those transactions. The master database knows what state each datafile is in, and it looks at each of them upon starting up. If it detects any that are corrupt, you have to restore those datafiles from your backup. The master database then looks at the datafile and realizes that it was restored from an earlier point in time. It then goes to the transaction log to “redo” all the transactions that have been recorded since that time. Uncommitted transactions are then rolled back. The actual order of this process differs from product to product. However, the main purpose remains the same. The rollback and transaction logs work together to ensure that all pages are returned to their proper state after a crash or reboot.

Many people have difficulty understanding the difference between the rollback log and the transaction log. Informix’s terminology helps in this case, because its terms *physical* and *logical* log illustrate exactly what the logs contain. The physical log contains a physical “image” of a page prior to it being changed by a transaction. It doesn’t know or care how the page was changed; it just knows what it looked like before it was changed. The logical log, on the other hand, keeps track of *how* the page was changed, so that the database can recreate this change after a recovery. Oracle’s terminology helps here as well. An undo or rollback log is the log that will allow you to “undo” or “remove” the changes done by a transaction (that were not yet committed). Redo logs (aka transaction logs) allow you to repeat or “redo” the committed transactions.

MySQL does have the binary log that contains SQL statements that can be replayed against a consistent database to redo those SQL statements. However, it is not used for crash recovery. Each storage engine has its own transaction log for this purpose.

Checkpoint

DB2	Exchange	Informix	MySQL	Oracle	PostgreSQL	SQL Server	Sybase
Checkpoint	Checkpoint	Checkpoint	Checkpoint	Checkpoint	Checkpoint	Checkpoint	Checkpoint

In order to increase performance, databases keep a lot of data in memory: recently changed pages, commonly accessed pages, before images of modified pages, and the transactions themselves. This means that if the system crashes at some point, some data will be lost, because RAM is volatile. The database needs some way to go back to a time that it knows

everything was on disk and nothing was in memory. This point in time is called the *checkpoint*.

At certain intervals, the database flushes everything to disk. All datafiles and logfiles are therefore in a consistent state. If the system were to crash without damaging the datafiles, the database would revert to this checkpoint, then replay any completed transactions that have been recorded since that checkpoint, and finally roll back any incomplete transactions. This ensures that the database can always be backed up in a consistent state.

---

[2] That is why I prefer the term “page” over the Oracle term, “block.” It helps to differentiate between filesystem blocks and database blocks, or pages. I have seen DBAs and SAs confuse each other talking about what block size a given Oracle database should have.

An Overview of a Page Change

We ask a lot of RDBMSs. We want them to be big and fast but to always ensure that the data is in a consistent state. To accomplish the speed requirement, they've got to keep lots of things in memory, but to ensure the integrity requirement, they've got to be very careful. It's difficult to juggle knives without cutting off a finger or two.

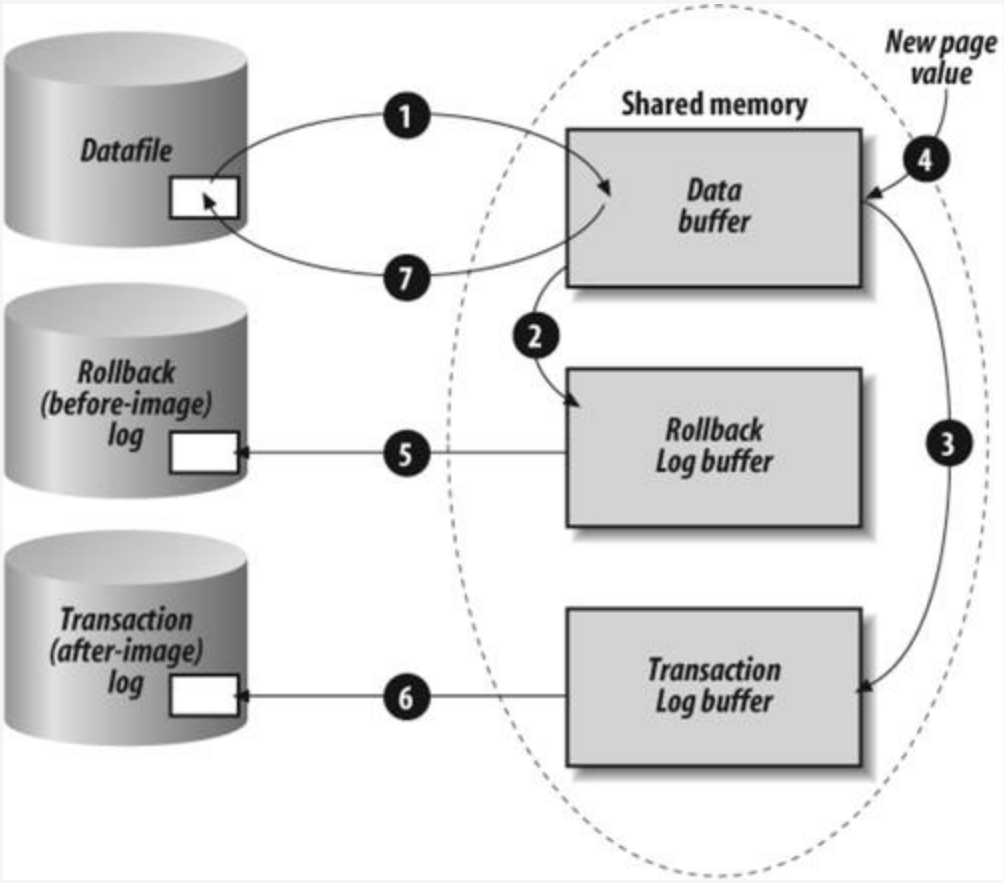


Figure 15-4. Anatomy of a page change

Please refer to Figure 15-4 throughout this section. It shows that there are four main storage areas of an RDBMS. There is the datafile itself, the rollback log, the transaction log, and shared memory. Shared memory is further divided into the data buffer, the rollback log buffer, and the transaction log buffer. (Other elements could be in shared

memory as well. Only relevant elements are listed here.) Not all RDBMSs work exactly this way, but they are all similar. What happens when a user enters a transaction? Before anything happens, the transaction and what it's going to change is recorded in the transaction log. Then, for each page that needs to be changed, the following steps are taken (note that the list's numbers relate to the numbers in the figure):

1. The RDBMS reads the page that is to be modified and loads it into the data buffer. Until proper safeguards have been put in place, this is where the page stays for a while.
2. The transaction then wants to change the page, but first it saves a copy of what it looked like prior to changing it. This before image is placed in the rollback log buffer.
3. The after image of the page is recorded in the transaction log buffer.
4. The transaction can now change the actual page that it wants to change. This change is made only to the copy of the page that is stored in the data buffer. Recording the change in the transaction log buffer before actually making the change is a safety measure.
5. Notice that up to this point, no changes have been made to disk. If the system were to crash right now, the disks would have no record of this transaction. Before the system can make any changes on disk, it needs to make sure the before image is safe. This is because it will need that before image to roll back this change if it doesn't complete due to a canceled transaction or a rebooted system. To ensure that the before image will be available, it flushes the rollback log buffer to the *physical* rollback log (the log on disk).

6. The after image of the page is now flushed to the physical transaction log. This image will be necessary if the transaction needs to be redone.
7. Now that the system has safely preserved the before image and after image of the page to be changed, it can flush the changed page to disk.

At some point, the transaction is committed. This fact is immediately recorded in the transaction log. That way, the system knows that the entire transaction completed, and should redo it in case of a system crash. This process ensures that no matter what time the system crashes (and it *will* crash), the system will remain in a consistent state.

ACID Compliance

The ACID (atomicity, consistency, isolation, and durability) model establishes four goals that every RDMBS must achieve if it wants to be ACID-compliant. A database that is not ACID-compliant will not be as reliable or resilient as an ACID-compliant database. Following are explanations of these four goals:

Atomicity

: Changes to a database are “all or nothing.” No matter what happens, a transaction is either a complete success or a complete failure.

Consistency

: No transactions will be allowed that will place a database into an inconsistent state. If any transaction violates a given database’s consistency rules, it will be rolled back.

Isolation

: Transactions must not interfere with each other. For example, one transaction should not read intermediate data created by another transaction. This is primarily accomplished through logging.

Durability

: Transactions committed to the database will not be lost for any reasons. This is what backups and transaction logs are for.

What Can Happen to an RDBMS?

A lot of things can happen to interfere with the normal operation of a database. What you need to do to get the database running again will depend on what broke it in the first place. The following is a list of some of the things that can happen to an RDBMS:

Device ownership change

: Someone can accidentally change the ownership of the raw devices or files that the database is using for datafiles. Since the database can no longer write to the files, it will cease to function. You will need to return the device to its proper ownership and possibly restore data.

Device permissions change

: This is similar to an ownership change, because the database engine can no longer write to the file. The fix is the same as the previous one.

Disk destruction or failure

: The only real protection against a bad disk is RAID or some type of backup or replication.

Logical corruption

: A database can become logically corrupted in all sorts of ways. Deleting an important table is an example of logical corruption. The process of retrieving and storing pages can also result in logical corruption that can only be seen using a database consistency checker or an export.

Controller goes bad

: If you are mirroring some of your devices, you should set up the mirroring so that a device on one SCSI controller

is mirrored to a device on another SCSI controller. This ensures that if a SCSI controller does go bad it won't take out both mirrors at once.

Assign a database raw device as a swap or filesystem

This one's a bad one. This is where proper documentation comes in handy. It also helps if your administrators are trained to look at the ownership of a device before they use it. If it is owned by someone other than root, then don't use it. To recover from this, you need to undo the change and restore from backup.

Another application using the raw device as mirror

This is similar to the preceding scenario; an administrator tries to use one of the database disks for something other than the database. Again, you will have to undo the change and restore from backup.

Backing Up an RDBMS

Protecting an RDBMS is very complex. There are several storage elements, including datafiles, rollback logs, transaction logs, and the master database. How do you get all of the data to a secondary storage medium if it's changing all the time?

> **Tip ✎**
>
> This chapter focuses primarily on how to back up the data in the database. We assume that you are also backing up other important information, such as the operating system, the database application, and the schema of the database.

Physical and Logical Backups

There are two primary methods of backing up an RDBMS database: physical backups and logical backups.

A *physical backup* physically backs up the datafiles. This is also referred to as a database backup. There are two types of physical backups, a cold backup and a hot backup. A *cold backup* is done by shutting down the database prior to doing the backup. This is often the simplest way to do a database backup, especially if your database's datafiles reside in the filesystem. All you have to do is shut down the database and run your normal filesystem backup utility. Unfortunately, this method may require your database to be shut down for a long time. That is why more and more environments are performing *hot backups*, which are done while the database is online. These types of backup, of course, require a lot more work behind the scenes, because

you are trying to copy the datafiles while the database is writing to them. You'll also need a backup utility that understands the internal structure of the database.

There are several ways to do physical backups:

- With most databases that use cooked files as datafiles, you can simply shut down the database and do a full system backup. Since everything exists as a regular filesystem file, everything will get backed up. If you need to restore, you can then restore the entire database from this backup. You then need to replay the transaction logs against the old database files. Many databases don't allow you to replay transaction logs against a backup that was made this way. For those databases (which include DB2, Sybase, SQL Server, and Sybase), this method will not work.
- If your datafiles are raw partitions in Unix, you can shut down the database and back them up if you have a utility or script to do so. For example, in Unix you use `dd`. This method is quite a bit more complex than the first, though, because you need to know which devices to run `dd` against.
- The next method of backing up a database is to back it up live to disk or tape using a utility provided for that purpose. Oracle uses `rman` for this purpose, DB2 uses the `db2 backup` command, Sybase and SQL Server provide the `dump` utility, and you can use `ntbackup` with Exchange.
- Another method of backing up databases is to use a utility that sends one or more streams of data to a commercial storage manager (i.e., backup software). This is the cleanest method if you can afford it (it can cost several thousand dollars per system). Each of the major database vendors provides an API that third-

party storage managers can write to that allows them to back up data from a database.

- A few commercial utilities provide functionality that is different from that of the previously listed options. The most popular of these is SQL Backtrack, which is now sold by BMC software. These utilities wrap around some of the native utilities and do not use the vendor-supplied API. They also enable you to send a data stream to commercial backup products, and some products have interfaces to the utilities. Still other options include interfaces to products that do not use the newer interfaces. The validity of backup and recovery programs that do not use the vendor-supplied API is left as a decision for the reader.

A *logical backup* copies, or exports, data objects (usually tables) but does not record the data's location. A logical backup can restore a deleted table without having to restore all the datafiles in which it resides. It also can be used to move a table from one database to another. These options are made possible by the definition of a logical backup: it backs up the data and not the data's location. Therefore, it can be restored into any location. Logical backups, however, do not have the ability to do a point-in-time recovery. They also can introduce referential integrity problems, because you could load a table that requires information from another table that is not present. The biggest problem with exports, though, is that they almost always need to be done with the database offline.

Logical backups are actually much simpler than physical backups. Each database has an export utility that creates a logical backup of one or more database objects to a file. Some of the commercial utilities also allow you to integrate logical and physical backups into your backup system.

Raw Devices Versus Cooked Files

This is an often-debated topic in DBA circles. Many DBAs want to use raw devices because they believe they are faster than using filesystem files. Some DBAs prefer to use filesystem (cooked) files due to their ease of administration.

The historical reasons for using raw partitions were increased performance and data integrity. A database spends a lot of time trying to ensure all the data is in its proper state. It logs every changed page and knows the exact condition of every page at any point in time. It assumes that when it tells the OS to write a page to the datafile, the OS does so. However, many filesystems cache writes to increase performance. This means that the database thinks that a page has been modified on disk when it really hasn't. This is a bad thing.

Modern databases resolve this problem by opening a cooked file in `O_SYNC` mode, which automatically flushes any changes from the OS buffer to the disk. This means that this historical argument is no longer true with most databases. Therefore, the only remaining reason for using raw partitions as datafiles is performance. Many claim a 5 to 15 percent performance increase over filesystem files. Your mileage may vary; these tests were performed on a closed track with a professional driver; *please do* try this at home. Find out what works best for you.

Get Every Instance

Chapter 2 includes a section called "Are You Backing Up What You Think You're Backing Up?" It talks about how your backup programs should be written so that everything in your system is automatically discovered and backed up.

If you add a new filesystem, you should not then have to edit your backup scripts to get it backed up. This goes double for databases because they often are added and deleted much more frequently than filesystems.

You need some way to ensure that every database instance on every server is being backed up. Commercial backup products can automatically ask the operating system which drives and filesystems it has. Wouldn't it be nice if you could do that with databases? Let's start with Oracle and Sybase. Sybase has the *interfaces* file that lists every server on each system. If an instance is not listed in this file, users cannot connect to it. Oracle has the *oratab* file that accomplishes the same task, but its use is not mandatory, as Sybase's *interfaces* file is. Programs can ask the Windows registry about which Exchange or SQL Server databases are running on a given server. DB2 DBAs can query its catalog. Unfortunately, Informix does not have this type of functionality unless you create it yourself.

Since the files described here are not always used and yet should be, I'd like to emphasize what I just said: *you need a centralized file that lists all the instances on the server*. You then should start with that file to determine what instances are on a given server—instances that need to be backed up. Enforce the use of the *oratab* file by writing startup scripts that start up only instances that are listed in those files. A wayward (or busy) DBA still can create an instance without putting it in this file and can even get it running. However, if you reboot the box enough times, the DBA will be sure to put it in the startup file so he doesn't have to manually start it every time! For Informix, create a file similar to oratab and write startup scripts that only start up databases listed in that file.

Transaction Log Dumps Are Not Incremental Backups

This important topic often is misunderstood. The confusion comes from Sybase and SQL Server documentation that often refers to a transaction log dump as an incremental backup. They are not the same thing!

What is the difference between the two? An *incremental backup* is a special backup that contains only the changed pages (blocks) since the last full (or higher level incremental) backup. A *transaction log dump* is a backup of all the *transactions* that have occurred since the last transaction log dump. They may sound similar, but they're not. The latter is much more difficult to manage and much slower to read.

Perhaps the best way to illustrate this would be to discuss Oracle's `rman` program, which has both incremental and transaction log backups. Suppose that you created a full backup on Friday. During the week, you did not perform any full backups and ran only redo log (transaction log) backups. Now suppose that it is Thursday, and you need to restore your database. You would have your full backup from Friday and your redo log backups from each day. To restore, you need to read your full backup and then read each of the redo logfiles in order. Assuming you made one backup volume per day, you would need seven volumes.

Now assume the same scenario as the preceding, except that you also ran an incremental backup every night. If you have to restore the database on Thursday, you would need the full backup from Friday, the latest incremental backup volume (presumably Wednesday's), and the redo logs for Thursday; in other words, three volumes instead of seven. That is because the incremental backup contains all changes since the full backup.

Besides the difference in complexity, reading an incremental backup also is much quicker than reading a transaction log backup. Ask anyone who has rolled through several days' worth of transaction logs. In one benchmark that I performed, reading two weeks of transaction logs took 36 hours. Reading an incremental backup covering the same time period took only one hour. The reason for this is simple. A given page may be changed several times; replaying the transaction log also changes it several times. Loading a true incremental backup changes it only once, to its last value.

DB2 has *incremental backups*, which back up all changes since the last full backup, and *delta backups*, which back up only changes since the last backup of any kind. They also have transaction log backups. Oracle and Informix support multiple levels in their backups (e.g., level 0, level 1, etc.).

Do It Yourself: Creating Your Own Backup Utility

You don't have to use a high-priced commercial utility to back up your databases. Doing so certainly can make your backups more automated or centrally controlled, but since most commercial utilities can cost thousands of dollars per system, many people are using homegrown systems.

Intermediary disk

This is one of the most popular ways to do homegrown database backups. It's fast, clean, and easy. The basic idea uses a script that backs up the database to disk. That backup is treated as a regular file and backed up by the nightly filesystem backup. You can even save the amount of disk space needed by compressing the files after they're

backed up. If you're really pressed for space, you can use named pipes to compress the backup *as* it's being written. In that case, you would need a backup disk that is only one-third to one-half the size of your original database disk (depending on the compression rate you get). Unless you have a very large database, this probably is cheaper than buying a commercial utility to perform this task. Each of the vendor-specific database backup chapters in this book explain how to do this.

Dedicated tape drive

You can use homegrown backup scripts to back up to a dedicated tape drive. This is a little more complicated and requires somewhat more work on your part. Depending on the size of your database, this may be more or less expensive than backing up to disk, but it definitely will be slower. It is also more complex, because you must keep track of each volume and label it in such a way that you know which database was backed up to it. (If you back up to disk, this can be done by naming the backup file the same name as the database.)

Shell/batch scripts

I assume that you are doing the preceding backups using some sort of shell or batch script. Scripts are much better than having a simple `cron` or `at` entry that says: `back up databaseA to deviceB`. Scripts can do lots of error checking and can be told to do things such as notify the DBA if something is wrong.

Calling a Professional

This is one of the biggest growth markets within the backup product industry. Most commercial filesystem backup products now have interfaces to automatically back up your database to volumes that are managed by their product. It's really a beautiful thing, but it does come at a price!

Each major database vendor—including all those covered in this book—has a backup utility API that commercial backup products can program to that allows them to back up that database. (The commercial backup product is often called a *storage manager*.) On a high level, these backup utilities all work in essentially the same way. The database vendor's utility generates one or more backup streams via an API that storage managers can talk to. The companies that produce the storage managers then can write a utility that talks to their storage manager on one side, and the database backup utility's API on the other side. Although the database backup utilities come bundled with the database products, the commercial backup products' utilities cost several thousand dollars.

These backup utilities sometimes will not function without a storage manager. For example, Oracle's `rman` and Informix's `onbar` must have a storage manager to talk to in order to back up to tape, although `rman` can back up to a disk without a storage manager. In contrast, Sybase's and SQL Server's `dump` utility, DB2's `db2 backup`, and Exchange's `ntbackup` can back up to tape or disk without any storage manager present. Since Oracle and Informix didn't want to force their customers to buy a storage manager or the interface to their backup utility, they came up with a compromise. Both vendors bundled a free, stripped-down version of a storage manager that gives you enough functionality to be able to use `rman` and `onbar` to do backups. Figure 15-5 uses Oracle and this storage manager to illustrate the different pieces of the backup puzzle.

Oracle uses `rman` to interface with the database on one side and the storage manager's database module on the other side. The storage manager communicates with the backup media on one side and the database module on the other side. The storage manager then uses the database module to interface between the storage manager and `rman`. The backup data flows from the Oracle database, through `rman`, through the database module, through the storage manager, and to the backup media. Restores obviously flow in the opposite direction.



Figure 15-5. A typical commercial database backup utility setup

Utilities for each of the big databases have their own backup and recovery history.

DB2

`db2 backup` has a long history and has always kept backup and recovery in mind. You can restore databases created on different types of platforms, and the restore utility can also be used to restore backup images that were produced on a previous version of DB2 (up to two versions earlier) as long as the word size (32- or 64-bit) is the same. You can even back up a database created on Linux for zSeries (mainframe) and restore it to an AIX-based DB2 UDB server. DB2 has also recently added the `recovery` command, which helps simplify restores.

Exchange

Exchange is a special-purpose database and does not have a built-in backup command. However, there is an API that `ntbackup` interfaces with, so you can use `ntbackup` to back up and recover a live Exchange database. You can also back up and recover at the storage group level. If you want to do mailbox-level backups, you need to use other utilities.

Informix

Informix backups have historically been done with `tbtape` (now called `ontape`), which is a standalone backup command designed to back up to tape. `ontape` is simple, has incremental capabilities, also can back up to disk, and backs up the database live. Some of these features, which always were assumed by Informix users to be present in other database systems, are just now appearing in other products. Informix also has `onbar`, which is designed specifically to send a stream of backup data to a commercial product. Whichever command you use, you can recover individual `dbspaces`.

MySQL

How you back up a MySQL database depends on which storage engine you use. Each major storage engine has its own backup and recovery method.

Oracle

Oracle databases can be backed up with the Recovery Manager, or `rman`. `rman` has come a long way since its introduction and is a much easier way to back up and restore your database than what Oracle now calls *user-managed backups*.

PostgreSQL

There are two main ways to back up PostgreSQL databases. The first is equivalent to the user-managed hot backups in Oracle, where you run a script that puts the database in backup mode, then copies its files. The second is a command (`pg_dump` or `pg_dumpall`) that dumps the entire database into a large text or binary file.

Sybase

Sybase has come a long way in the backup arena. You can now recover an individual tablespace or device. Unfortunately, since many people use only the default segment or filegroup, this means most DBAs have to restore the entire database or nothing at all. Creating additional segments or filegroups could help with this issue.

SQL Server

The SQL Server backup utility provides basic backup and restore capabilities. It enables you to do incremental, differential, full, and transaction log backups without third-party tools.

Restoring an RDBMS

The process of restoring an RDBMS varies according to the backup methods that you used, of course. How you proceed to restore is based on the status of your nondata and data disks and whether you are able to do partial restores online.

Loss of Any Nondata Disk

A *data disk* is defined as any disk that contains a database object. (*Database object* and other terms are defined earlier in this chapter.) If you keep your database objects intact, you actually don’t need to restore the database, you’ve just got to restore all the parts that make it work! This can range from a restore of a single database setup file to a complete restore of everything on another system.

Executables

: Your database can’t work if its executables aren’t there. This part of the recovery is much simpler if you have all your executables located in a special filesystem.

Setup files

: In a pinch, you can restore the executables by copying them from a known good system, but that won’t work for your database setup files. Each instance often has an initialization file that sets up certain variables like the instance name and the location of the master database. These setup files usually can be recreated if you have good logs that tell you how you made them the first time, but it is probably easier to recover them from backup.

Customized OS files

Databases often require you to edit system configuration files such as */etc/system*. Changes to these files might include things like customizing shared memory or changing the TCP port over which software will communicate. If you are restoring onto a brand new install or onto another system, these changes will need to be repeated, and often changes to your OS files are forgotten or poorly documented. Unless you properly prepare for this situation, it's probably easier to just follow the installation instructions for a standard installation. If you're reading this in advance of such an outage, now is the time to find out what these changes are and document them. If you know what files are typically changed, you can even write a program that automatically documents them for you.

License setup files

Database products have not typically used heavy licensing enforcement systems, but this will probably change over time. If yours does use such a system (e.g., FlexLM), you need to restore these files before your database will function properly.

Loss of a Data Disk

The complexity and difficulty of your restore can vary greatly depending on which data disk you lost and how well you prepared in advance for such a loss.

Master database

The complete loss of a Sybase or SQL Server's master database, Informix `rootdbs`, DB2's catalog, or Oracle's control file is very difficult to recover from—so much so that you need to ensure that it never happens. (You'll be sorry if you don't!) Mirror your Sybase master database.

Mirror your DB2 catalog. Mirror your Oracle control files. Just do it. Even if you can’t afford enough disk to mirror anything else, mirror this. It doesn’t take up that much extra disk space, and the amount of time and frustration you will save yourself is immense. This is the easiest thing you can do to save huge amounts of time in a major restore.

Other databases

Unless you are using Oracle (which has a one-to-one database-to-instance relationship), you may have multiple databases within an instance. If you lose only one device within a database (other than the master database), recovering from this is not too bad. You also probably can leave the rest of the instance and any other databases online while you are doing the restore. The best thing you can do to automate restoring single databases is to properly document where they are and what devices they consist of.

Backups

One of the most popular ways to back up a database is to save it to disk and then allow the filesystem backup program to put it onto a backup volume. This is a very efficient method for many reasons. However, it does have one downside. Suppose you lost a data disk and the disk on which you store the backups. You would first have to restore the disk backup file from the backup volume, then restore the database from the disk file. If this two-step procedure is required, it will take longer than recovering straight from a backup volume.

Transaction log backups

The same rule applies to the backups of your transaction logs. You will need every one of these that were made since the last full or incremental database backup.

Before you start a large restore, check the time that the backup was made and make sure that you have all the transaction log backups since then. If you don't, you can start restoring them before you actually need them.

Losing the Master Database

Recovering the master database is full of Catch-22s because a lot of configuration and status information is stored inside that database. This information can be recreated if you lose it, but it must be recreated manually. You must have a complete history of the instance, how it was put together, where the database file and logfiles are, and what status they are in. Then you have to tell the master database all the things that it should know already. This won't be too hard if you saved all this information in an easy-to-find location up front. If not, you're in for a long, painful recovery.

Warning ☠

If you do database backups infrequently, you may need a significant number of transaction log backups to complete the restore. While restoring them, you must be careful to have enough space to do so. You may have to restore them into an alternate location or compress them. You then can move (or uncompress) them a few at a time as the `restore` program asks for them.

Online transaction logs

This is where you can suffer data loss. Even if you have a good backup and all the transaction logs since the last backup, you can be in trouble if you lose the data with the online logs in it. This would be the disk with

Informix's logical log or Oracle's online redo logs. (Sybase, SQL Server, and DB2 store the online transaction log in the master database.) One way to prevent this tragedy is to mirror this log.

Online Partial Restores

Some databases allow you to bring part of the database online while leaving one tablespace or datafile offline. This partial availability may allow you more time to complete a difficult restore or reduce the overall impact to your user community. This is especially true if the portion of the database that you are restoring contains a table that is not accessed frequently. Before considering such a restore, though, consider these factors:

Interdependency of data within the database

If the table contains data that is not accessed frequently, this may be a perfect candidate for a partial restore. You also might consider such a restore if the rest of the database can function normally (or in a slightly diminished capacity) without this table. However, suppose the database is the sales database, and this table contains all the actual sales transactions. The rest of the database is fine (e.g., names, phone numbers, addresses), but the sole purpose of this database is to track these sales. Without this table of transaction data, the database is useless. Since doing a partial restore increases the overall restore time, doing a partial restore would be a bad idea in this case.

Physical relationship between tables and tablespaces

Most database backup products do not allow you to restore on the table level. They sometimes allow you to restore on the tablespace level, however. Suppose you

have lost one disk. You need to recover the tablespace that this disk resides in, right? Suppose that you had a partitioned table that resides on multiple tablespaces. That would mean that the entire table would be unavailable. If this table is not needed for normal operation, as described before, you might consider a partial restore. Again, however, remember that a partial restore increases your overall restore time.

Time requirement for a complete restore

Some environments don’t want to hear about partially functioning databases. “Tell me when it’s up. Don’t say it’s almost up or partially up, just tell me when you’re done!” These environments are more concerned with overall downtime and should be treated thusly. You need to restore the database in the fastest way possible. That probably would be to shut down the database and restore the affected tablespace.

Documentation and Testing

Restoring an RDBMS is the most complex procedure that you will ever do, and there is usually very little time in which to do it. The users want the database up now and don’t really care that you’ve never done this before. You need to document and test your database backup procedures often to make sure that they work. The following guidelines may help:

- Set up a standalone machine that does not have any other databases on it. This machine doesn’t have to be large or even dedicated to this purpose, but it would be good if you could reinstall the OS for a new test. This often can be done when you buy a new machine. Before you put it to work for real, do some test database restores on it.
- When you test your restores, test the worst-case scenario. Make sure that you know how to install the product onto a new system and that you know all of the files that you need to edit to make it work. This is why you should be doing this on a virgin operating system. That way, you will always have to edit */etc/system* on a Unix, for example, instead of forgetting it without penalty because it’s already been edited.
- You can set up a test instance and then back up and restore that. However, the best thing to do is to take a normal database backup from one of your production systems and try to restore it to the test system.
- Make the procedure detailed enough that anyone with good SA or DBA skills should be able to follow it. If possible, do not have the person who wrote the procedure perform the test. This is the perfect task to

hire a consultant for. See if someone who knows what they are doing but doesn't know your environment can follow the procedure.

- Have all your DBAs participate, whether they've got time or not! The worst thing that can happen is that only one or two people know how to restore the databases. Then when a database crashes, one of them is sick and the other one just quit or is on vacation in the Bahamas. (I once saw a restore go on for hours because the DBA didn't know he had to press Enter! He called me saying, "Man, this thing is taking forever!") Make sure every DBA knows how to restore your databases!

Unique Database Requirements

The unique backup and recovery requirements of each of the databases covered in this book could generate an entire book for each product. What, then, is the purpose of these chapters? That's simple—to break down, once and for all, the "Berlin Wall" that exists between DBAs and SAs. Until now, if you wanted to learn about how to back up DB2, you had to buy a whole book on DB2. One of the problems with books like these is that they assume you are a DBA! The authors assume that you understand tablespaces and transaction logs and rollbacks. The result is that you get only half the story. This book, on the other hand, assumes only that you have read this and other chapters in the book.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 16. Oracle Backup and Recovery

Oracle is a very popular commercial relational database management system with a very large installed base. Oracle can be backed up using two very different methods, and each method has its advantages and disadvantages—and its strong supporters and detractors.

Two Backup Methods

The backup and recovery method with the most history is now called *user-managed backup*. A user-managed backup consists of putting the Oracle database into a "backup friendly" state, then backing up its files using whatever tool strikes your fancy. Once the files have been backed up, you can take the database out of its "backup friendly" state. User-managed backup is documented and supported, but is not what Oracle would prefer you do.

The preferred method of backing up Oracle databases is the Recovery Manager (`rman`), which was first available in Oracle 8. The `rman` utility can be used independently or with a commercial backup utility to back up Oracle to disk or tape. It offers a number of advantages over user-managed backups, including incremental backups, data integrity checks, block-level media recovery, and guided restores. Those who have learned `rman` swear by it, especially the new and improved version available in Oracle 10*g*.

Tip ✎

As of this writing, approximately half the Oracle community is using `rman` and half is performing user-managed backups. Both methods will be covered in this chapter, starting with this comparison of the two.

rman

`rman` has a number of advantages over user-managed backups. The first advantage is that `rman` will get much more research and development funds than user-managed backups. Where user-managed backups haven't changed much in the last several years, dozens of features have been added to `rman` in that time.

`rman` also has a number of features that aren't ever going to be available with user-managed backups:

Incremental backups

: `rman` can create backups that contain the blocks that have changed since the last full backup. The speed and performance of incremental backups are significantly enhanced in 10*g*.

Incrementally updating backups

: If you're using disk as the target for `rman`, it has the ability to take the latest incremental and merge it with the full backup already on disk, creating a new full backup without having to actually perform a full backup.

Data integrity checking

: Previously, you had to perform an export of Oracle to check the integrity of blocks on disk. `rman` now performs

this check for you as part of the backup.

Block-level media recovery of a corrupted datafile

: If `rman` identifies any corrupt blocks in a datafile, it can recover them one block at a time.

Directed restores/recoveries

: You tell `rman` to restore the database, and it figures out what needs to be restored and automatically restores it.

Many other features

: Oracle has continued to add many features to `rman`, none of which will be available with user-managed backups.

`rman` has two disadvantages when compared to user-managed backups: learning curve and cost. While user-managed backups aren't easy to learn either, they've been around a lot longer, and many DBAs understand them. Many DBAs still view `rman` with trepidation, often due to the large manual they'd have to read. The second disadvantage is that you must either back up to disk or purchase a third-party media manager to allow you to back up to tape. Some environments don't have the money for enough disk to store an entire copy of their database (even if compressed), but they do have a tape drive they can `dump`/`tar`/`cpio`/`ntbackup` to. If they don't have the money for more disk, they don't have the money for a commercial media manager to back up to tape.

> **Tip ✎**
>
> If you want the advantages that `rman` offers but don't want to purchase your commercial backup utility's interface to it, you can still integrate the two by using `rman` to back up to a disk, then using your backup utility to back up that disk. It's not perfect, but it's something.

User-Managed Backups

The biggest advantage user-managed backups have is history. Any DBA who has worked with Oracle for a long time probably understands them.

User-managed backups can also be relatively simple, especially if you can shut down the database to do a cold backup. All you have to do is shut it down prior to performing your usual filesystem backup, using whatever tool you want to use. If you can't shut down the database, put it in backup mode and back it up live. (Backup mode is covered later in the chapter.) Once you've run the backup, start up the database or take it out of backup mode.

This procedure allows you to integrate backups of Oracle with any backup utility without having to pay for its interface to `rman`. While `rman` is free, your backup utility's interface to it definitely won't be. These interfaces can cost several thousand dollars per server.

Tip ✎

This chapter uses the command `sqlplus /nolog`, followed by `connect / as sysdba` to connect to Oracle 9*i* and 10*g* databases. If you are running Oracle 8*i*, you can use `svrmgrl` and `connect internal`. (In the course of reading this chapter, you'll learn some really great reasons to upgrade to 10*g*.)

Oracle Architecture

As mentioned in Chapter 15, it is important to understand the design of the database that is being backed up. Therefore, this chapter starts with a discussion of Oracle architecture. Similar information is provided in Chapter 15, but this chapter concentrates on information specific to Oracle. Just as in Chapter 15, we start with the power user's view of the database, then continue with that of the database administrator. This chapter uses Oracle-specific terms. To see how a particular term relates to one used in DB2, SQL Server, or Sybase, consult Chapter 15. As much as possible, these architectural elements are presented in a "building block" order. Elements that are used to explain other elements are presented first. For example, we explain what a tablespace is before explaining what the `backup tablespace` command does.

The Power User's View

Unless a power user wants to start doing the DBA's job of putting a database together, the following terms should be all she needs to know. This view also could be called the "logical" view, because many of the elements described in this view don't exist in a physical sense.

Instance

An *instance* is a set of processes through which the Oracle database talks to shared memory and the shared memory that Oracle allocates for its use. Often, there is more than one instance on a single system. When an Oracle instance is opened, the database connected to it becomes available.

On a Unix/Linux system, an instance can be identified by a set of processes with the pattern `ora_` *`function_SID`*, where *`function`* is a string indicating which Oracle function the process applies to and *`SID`* is the instance name. On Windows, each Oracle SID has its own service named `OracleService` *`SID`*, where *`SID`* is the instance name.

On a Unix/Linux system, an instance is automatically started with the `dbstart` script and shut down with the `dbshut` script. You can use these scripts or SQL*Plus commands to manually stop and start Oracle. In Unix or Linux, you have to place this script into the appropriate directory and add the appropriate symbolic links to have it run automatically after a reboot. To automatically start and stop instances in Windows, you should enable automatic start in the Control Panel for the appropriate OracleService. (If you cannot find `OracleService` *`SID`* in the Control Panel, you can create it using the `oradim` utility.) You then need to tell Oracle to automatically start the database when the service is started. You can do this by right-clicking on the SID in the Administrative Assistant; choosing Startup/Shutdown Options; choosing the Oracle Instance tab; and selecting “Start up instance when service is started,” “Shut down instance when service is stopped,” or both.

Database

The *database* is what most people think about when they are using Oracle. That’s because the database contains the data! A database is a collection of files that contain tables, indexes, and other important database objects. Unless you’re using Oracle Real Application Clusters (RAC), there is a one-to-one relationship between instances and databases. Without Oracle RAC, a database connects to only one instance, and an instance mounts only one

database. RACs run multiple instances (most likely on multiple servers) that share a single database.

That is why Oracle DBAs often use the two terms interchangeably. Technically, though, the *instance* is a set of processes through which the database talks to Oracle's shared memory segments whereas the *database* is the collection of files that contain the data.

Table

A *table* is a collection of related rows that all have the same attributes. There are three types of tables in Oracle: relational, object, and XML.

Index

A database *index* is analogous to an index in a book: it allows Oracle to find data quickly. Again, an Oracle index is the same as anyone else's index, and it presents no unique backup requirements. An index is a *derived* object; it is created based on the attributes in another table so that it can be recreated during a restore, and it's usually faster to recreate an index than to restore it. Oracle has several types of indexes, including normal, bitmap, partitioned, function-based, and domain indexes.

Large object datatypes

Oracle 8 has special datatypes called BLOB, CLOB, and BFILE for storing large objects such as text or graphics. The BLOB and CLOB datatypes present no special backup requirements because they are stored within the database itself. (A BLOB typically contains image files, and a CLOB normally contains text data.) However, the BFILE datatype stores only a pointer inside the database to a file that

actually resides somewhere in a filesystem on the database server. This aspect requires some special attention during backups. See the section “LOB space” in Chapter 15 for more information.

Object

An *object* is any type of database object, such as a table or an index; it’s really a generic term rather than an Oracle-specific term. Unfortunately, Oracle also uses the term “object” to refer to reusable components created by object-oriented SQL programming. Here, it is used simply as a generic way to refer to any type of table, index, or other entity within Oracle.

Row

A *row* is a collection of related attributes, such as all the information about a specific customer. Oracle also may refer to this as a *record*.

Attribute

An *attribute* is any specific value (also known as a “column” or “field”) within a row.

The DBA’s View

Now that we have covered the logical structure of an Oracle database, let’s concentrate on the physical structure. Since only the DBA should need to know this information, we will call it “the DBA’s view.”

Blocks

A *block* is the smallest piece of data that can be moved within the database. Oracle allows a custom block size for each instance; the size can range from 2 K to 32 K, and each tablespace (defined later) can have its own block size. A block is what is referred to as a *page* in other RDBMSs.

Extents

An *extent* is a collection of contiguous Oracle blocks that are treated as one unit. The size of an extent varies based on a combination of factors, the most important of which is the tablespace storage allocation method defined at tablespace creation.

Segment

A *segment* is the collection of extents dedicated to a database object. (An object is a table, index, etc.) Depending on the type of object, extents may be allocated or taken away to meet the storage needs of a given table. Oracle8 had the rollback segment, but this has been replaced in later versions with undo segments, which are stored in an undo tablespace. (See the later sections "Tablespace" and "Undo tablespace.")

Datafile

An Oracle *datafile* can be stored on either a raw (disk device) or cooked (filesystem) file. Once they are created, the syntax to work with raw and cooked datafiles is the same. However, if `rman` is not used to back up the Oracle datafiles, backup scripts do have to take the type of datafile into account. If the backup script is going to support datafiles on raw partitions, it will need to use `dd` or some other command that can back up a raw partition. Using `cp`

or `tar` will not work because they support only filesystem files.

Each Oracle datafile contains a special header block that holds that datafile's system change number (SCN). Each transaction is assigned an SCN, the SCN in each datafile is updated every time a change is made to that datafile, and the control file keeps track of the current SCN. When an instance is started, the current SCN is checked against the SCN markers in each datafile. (See the section "Control file" later in this chapter.)

> **Tip ✎**
>
> Please see an important explanation of the role that the SCN plays during hot backup in the section "Debunking Hot-Backup Myths" later in this chapter.

Tablespace

The *tablespace* is the virtual area into which a DBA creates tables and other objects. It consists of one or more datafiles and is created by the `create tablespace` *`tablespace_name`* `datafile` *`device`* command. A tablespace may contain several tables. The space that each object (e.g., table) occupies within that tablespace is a segment (see the earlier definition of segment).

As of 10*g*, every Oracle database has at least three tablespaces: system, sysaux, and undo. The *system* tablespace stores the data dictionary, PL/SQL programs, view definitions, and other types of instance-wide information. The *sysaux* tablespace stores non-system-related tables and indexes that traditionally were placed in the system tablespace. Its name implies that it is the *auxiliary system* tablespace. The *undo* tablespace contains the

undo segments, which replaced rollback segments. When it comes to backup and recovery, the main difference between these tablespaces and the rest of the tablespaces is that they must be recovered offline. That is because the instance cannot be brought online without these tablespaces. Other tablespaces can be recovered after the instance has been brought online.

Partition

A table or index can be divided into chunks called *partitions* and spread across multiple tablespaces for performance and availability reasons. As long as you are backing up all tablespaces, partitioned tables do not present any challenges to backup and recovery.

Control file

The *control file* is a database (of sorts) that keeps track of the status of all the files that comprise a given database and maintains `rman` backup metadata. It knows about all tablespaces, datafiles, archive logs, and redo logs within the database. It also knows the current state of each of these files by tracking each object's SCN. Every transaction is assigned an SCN, and every time a change is made, the SCN gets updated both in the control file and in each datafile. (See the earlier section "Datafile.") That way, when the instance is starting up, the control file has a record of what SCN the file should be at, and it checks that against the SCN that the file has. This is how it "notices" that a file is older than the control file and in need of media recovery. Also, if an older control file is put in place, Oracle will see that the SCN of the datafiles is higher than those that it has recorded in the control file. That's when Oracle displays the `datafile is more recent than the controlfile` error.

If you're performing user-managed backups, control files should be backed up using both the `backup controlfile to filename` and `backup controlfile to trace` commands. If you're using `rman`, you can use the `backup controlfile` command, or you can have `rman` automatically back up the control file every time you run a backup by setting the `controlfile autobackup` parameter to `on`. You should also issue the `backup controlfile to trace` command within `sqlplus`. Additionally, it's a good practice to copy the results of this command to a known location to make it easy to find during recovery. There is a scenario that we will cover in the "Recovering Oracle" section where this trace output will come in quite handy. Restoring control files is a bit tricky. The mechanics of restoring control files are covered later in the section "Recovering Oracle".

> **Tip ✎**
>
> Back up the control file to a file or within `rman`, and back it up to trace, too. Both backups can be very useful.

It is best to avoid having to restore or rebuild a control file. The best way to do this is to mirror your control files within Oracle via a function Oracle calls *multiplexing*. This allows you to create two or three copies of the control file, each of which is written to every time the control file is updated.

> **Tip ✎**
>
> Please don’t confuse the Oracle term *multiplexing* with how this term is used everywhere else in the backup and recovery space (the sending of simultaneous backup streams to a single tape drive). To minimize this confusion, this chapter refers to multiplexing in Oracle as multiplexing/mirroring or multiplexed/mirrored.

Multiplexing/mirroring is slightly different from disk-level mirroring and offers an additional level of protection. With disk-level mirroring, you have one control file that’s mirrored to multiple disks via RAID 1. This protects you from disk failure, but not human error. If someone accidentally deleted the control file, it would instantly be deleted on each disk it was mirrored on. Storing multiplexed/mirrored control files on separate disks protects you from disk failure as well as accidental deletion. If a single control file gets deleted or corrupted, the instance becomes inoperable. However, the deleted/corrupted control file can easily be restored via one of the multiplexed copies, and the instance can be returned to proper operation.

> **Tip ✎**
>
> Make sure you are multiplexing/mirroring your control files to separate disks. They take up very little space and provide an incredible amount of recovery flexibility.

Transaction

A *transaction* is any activity by a user or a DBA that changes one or more attributes in an Oracle database. (A set of

commands that ends with a `commit` statement is treated as one transaction.) Logically, a transaction modifies one or more attributes, but what eventually occurs physically is a modification to one or more blocks within the Oracle database.

Undo tablespace

Undo data is now used for three purposes, the first of which is to undo an aborted transaction or a transaction that was not yet committed when the database crashed. Undo information also provides a consistent read for long-running queries. (See the section "ACID Compliance" in Chapter 15.) Finally, undo information can be used by Oracle's flashback feature, which allows you to manually undo transactions. For these reasons, it's important to keep undo information as long as possible.

Prior to 9*i*, undo was managed using rollback segments. The biggest challenge with rollback segments was trying to figure out how large they should be. If you made them too large, you wasted space. If you made them too small, you would get the dreaded `ORA-01555 snapshot too old` error, indicating that a long-running query was not able to obtain the consistent read it needed. If you had a very active database with a lot of long transactions or long running queries, this could prove to be a real problem.

Oracle 9*i* introduced automatic undo management, or AUM. You can continue to use manual undo management using rollback segments, but once you understand the value of AUM, it's hard to understand why you wouldn't use it. When configuring an Oracle database, you now specify an undo tablespace. Oracle automatically creates undo segments in this tablespace instead of requiring you to manually create rollback segments. AUM eliminates the complexities of managing rollback segment space and lets

you exert control over how long undo is retained before being overwritten. Oracle strongly recommends that you use undo tablespaces to manage undo rather than rollback segments.

At first, it's hard to differentiate between a tablespace dedicated to rollback segments and a tablespace dedicated to undo segments. Perhaps an explanation of how Oracle manages the automatic deletion of older undo data will help.

You can specify a minimum amount of time that undo information is to be kept using the `undo_retention` parameter. When Oracle needs space for more undo data, Oracle does its best to honor that value by deleting the oldest expired undo data first. However, if the undo tablespace is out of space, Oracle may begin deleting unexpired undo information, which unfortunately can result in the same `snapshot too old` error that you could experience with rollback segments. However, if you enable `autoextend` on the undo tablespace, the tablespace automatically extends to the maximum size you have specified. This automatic space management is what makes AUM so popular with those who have used it.

Checkpoint

A *checkpoint* is the point at which all changed data that is kept in memory is flushed to disk. In Oracle, a DBA can force a checkpoint with the `alter system checkpoint` command, but a checkpoint also is done automatically every time the database switches to a different online redo log group.

Flash recovery area

The components that create different backup and recovery-related files (e.g., `rman`, `alter database backup controlfile`, `alter system switch logfile`) have no knowledge of each other or of the space available in the filesystem in which they're storing data.

One of the big reasons to upgrade to Oracle 10*g* is its flash recovery area, which solves this problem by automating management of backup-related files. Choose a location on disk and an upper limit for storage space, and set a retention policy that governs how long backup files are needed for recovery, and the database automatically manages the storage used for backups, archived redo logs, and other recovery-related files for your database within that space. To create a flash recovery area, specify a value for the `db_recovery_file_dest` and `db_recovery_file_size` parameters in your startup file. To have archive logs sent to the flash recovery area, specify `location = use_db_recovery_file_dest` as the value for one of your `log_archive_dest_` *n* parameters.

Redo log

If the undo tablespace or rollback segments contain the information to roll back a transaction, the *redo log* contains the information to "roll forward" transactions. Every time that Oracle needs to change a block on disk, it records the change vector in the redo log; that is, it records how it changed the block, not the value it changed it to. A mathematical explanation may be helpful here. Suppose that you had a variable with a value of 100 and added 1 to it. To record the change vector, you would record +1; to record the changed value, you would record 101. This is how Oracle records information to the redo logs during normal operation. As explained in the later section "Debunking Hot-Backup Myths," it switches to recording

the changed value when a tablespace is in hot-backup mode.

In times of recovery, the redo log is used to redo (or replay) transactions that have occurred since the last checkpoint or since the backup that is being used for a restore. Oracle can be configured to use both online redo logs and offline (archived) redo logs.

> **Tip ✎**
>
> The online and archived redo logs are essential to recovering from a crash or disk failure. Learn everything you can about how they work and protect them as if they were gold!

Originally, the online redo logs were a few (typically three) files to which Oracle wrote the logs of each transaction. The problem with this approach is that the log to which Oracle was currently writing always contained the only copy of the most recent transaction logs. If the disk that this log was stored on were to crash, Oracle would not be able to recover up to the point of failure. This is why, in addition to multiplexing/mirroring the control file, Oracle also supports multiplexing/mirroring redo logs as well. Instead of using individual redo logs, you can now write redo information to a log group. A *log group* is a set of one or more files that are written to simultaneously by Oracle—essentially a mirror for the redo logs. Believe it or not, you're only required to have one member of each log group. Obviously, if you don't have multiple members of the log group, though, you're not helping yourself much. The usual practice is to put three members of each log group on three separate disks—often the same disks that you're multiplexing your control files to. The separate files within a log group are referred to as members. Each log group is

treated as a single logfile, and all transaction records are simultaneously written to all disks within the currently active log group.

Now, instead of three or more separate files, any one of which could render the database useless if damaged, there are three or more separate log groups of multiplexed/mirrored files. If each log group is assigned more than one member, every transaction is being recorded in more than one place. After a crash, Oracle can read any one of these members to perform crash recovery.

Oracle writes to the log groups in a cyclical fashion. It writes to one log group until that log group is full. It then performs a log switch and starts writing to the next log group. As soon as this happens, the log group that was just filled is then copied to an archived redo logfile, if running in `archivelogmode` and automatic archiving is enabled. If `archivelogmode` is not enabled, this file is not copied and is simply overwritten the next time that Oracle needs to write to that log. If automatic archiving is not enabled, the instance hangs the next time Oracle needs to write to an unarchived redo log.

For nonenterprise customers, each of the online redo logs is copied to the filename pattern specified by the value in one or more `log_archive_dest_` *n* parameters in the parameter file, followed by an incremented string specified by the `log_archive_format` parameter in the parameter file. For example, assume that `log_archive_dest_0` is set to */archivelogs/arch* and `log_archive_format` is set to *%s* `.log`, where *%s* is Oracle's variable for the current sequence number. If the current sequence number is 293, a listing of the *archivelogs* directory might show the following:

```
# cd /archivelogs/arch
# ls -l arch*
arch291.log
```

```
arch292.log
arch293.log
```

The `log_archive_dest_`*n* parameter is an enhanced version of the `log_archive_dest` parameter. Where `log_archive_dest` could be set to only one destination, you can have multiple `log_archive_dest_`*n* parameters, each of which can be set to a directory or to the flash recovery area by specifying `db_recovery_file_dest`. (See the section "Flash recovery area" earlier in this chapter.)

Depending on how much activity a database has, the archive log destination directory may have hundreds of files over time. If you send archive logs directly to a directory, Oracle does not manage this area; however, if you're sending archive logs to the flash recovery area, Oracle manages the space for you. If you're managing the space, a `cron` job must be set up to clean up archive log destinations. As long as these files are being backed up to some kind of backup media, they can be removed after a few days. However, the more logs there are on disk, the better off the database will be because it may sometimes be necessary to restore from a backup that is not the most current. (For example, this could happen if the current backup volume is damaged.) If all the archive logs since the time the old backup was taken are online, it's a breeze. If they aren't, they have to be restored as well. That can create an available-space problem for many environments. This is why I recommend having enough space to store enough archive logs to span two backup cycles. For example, if the system does a full database backup once a night, there should be enough space to have at least two days' worth of redo logs online. If it backs up once a week, there should be enough storage for two weeks' worth of transaction logs. (This is yet another reason for backing up every night.)

In summary, the online redo logs are usually three or more log groups that Oracle cycles through to write the current transaction log data. A log group is a set of one or more logs that Oracle treats as one redo log. Log groups should have more than one member; more than one member means there is little chance for data corruption in case of disk failure. Once Oracle fills up one online redo log group, it copies that redo log to the archive log destination as a separate file with a sequence number contained in the filename. It makes this copy only if you are running in `archivelog` mode and automatic archiving is enabled.

Initialization parameters

There are a number of *initialization parameters* in Oracle that are important to know during recovery, so it's vital to learn where those parameters are stored both in and outside of the database.

Historically, these initialization parameters were stored in a text file named *init<SID>.ora*. You could change most of the parameters inside the database, but if you wanted them to survive a reboot, you also had to make the changes in the *init<SID>.ora* file.

Oracle 9*i* introduced the concept of a *server parameter file*, or *spfile* for short. An spfile is a binary file named *spfile<SID>.ora* that is usually stored in the *ORACLE_HOME/dbs* directory. It cannot be edited, but it can be directly controlled by Oracle, allowing dynamic configuration changes to be written to the spfile automatically so that they will survive a reboot. For those who are used to *init<SID>.ora* files, an spfile can be a bit of an adjustment because there's nothing for you to edit. There are actually a few ways to change initialization parameters.

If the database is running, simply set the appropriate parameter via the `alter system set` *`parameter_name`* = *`value`* command. This action immediately changes the parameter in the running database and also stores it in the spfile. This is certainly easier than the old method of changing it in the database and the *init.ora* file. Some parameters, such as `log_archive_start`, cannot be changed this way. You must change them in the spfile and then restart.

> **Tip ✎**
>
> You can query both the currently used parameters and the parameters stored in the parameter file using *v$parameter* and *v$sparameter* tables, respectively.

If you want to change a parameter in the spfile without using the database to do it, you need to create a text version of the file, edit it, then import the text file into a new spfile. The following SQL command creates a pfile called *pfilename* from the current spfile:

```
SQL> create pfile='pfilename' from spfile
```

Once you've edited *pfilename* and made the appropriate changes, you can make a new spfile from the pfile using this SQL command:

```
SQL> create spfile from pfile='pfilename'
```

> **Tip ✎**
>
> Although you can query the *v$parameter* or *v$sparameter* tables even if a database is down, it might be helpful to occasionally make a text export of your spfile to allow you to look at it during a recovery.

Restore versus recover

Oracle distinguishes between a *restore* and a *recovery*. A *restore* is meant in the traditional sense of the term; it returns the datafile to the way it looked when you last backed it up by reading that backup from tape or disk. You then initiate a *recovery*, which applies redo against that datafile to bring it up to the current point in time. This is also referred to as *media recovery*.

Finding All Instances

All procedures in this chapter assume you know what the instances are on your server or that you're able to determine what they are. Unfortunately, there is no foolproof way to determine all instances on all machines. For example, I could tell you to use the *oratab* file on Unix, but not everybody uses it, and it can be out of date. For Windows, I could tell you to list the registry keys matching a certain pattern, but like the `oratab` file, these keys do not always represent real instances. Another method would be to list the instances that are actually running on the machine. This finds active instances, of course, but it won't find instances that were not running when the backup ran.

The best you can do is to tell everybody the method you're using to determine the list of instances, then stick with it. Tell them that if something's in *oratab* or listed in the Windows Registry, it gets backed up. If it's not there, it won't get backed up. Tell them that you're going to back up any running instances and will not back up instances that were manually shut down. Just make a choice, and publish it well.

You can enforce the use of the *oratab* file on Unix/Linux or the registry keys in Windows by automatically starting only

databases found there and taking note any time someone complains about an instance that didn't start after a reboot. Explain to them that it needs to be in the registry or *oratab*. If an instance is not in the *oratab*, it is not automatically started, and it doesn't get backed up!

The *oratab* file is usually located in */etc/oratab* or */var/oracle/oratab*. In Windows, you want to look at the following registry tree:

```
\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEn
```

Usually *n* is `0`, but there can be additional registry trees with other digits if there are additional `ORACLE_HOME`s configured on this machine. Underneath this registry tree, you should find one or more of the following values:

- `ORA_`*`SID`*`_AUTOSTART`
- `ORA_`*`SID`*`_PFILE`
- `ORA_`*`SID`*`_SHUTDOWN`

You can then parse that list to determine a list of Oracle SIDs.

> **Tip ✎**
>
> If you want to use the *oratab*/registry method but also want to double check that you're getting everything, you can also use the process list method discussed in the following paragraphs to see if any instances are running that aren't in the *oratab* or registry.

One way to do this is just to list the instances that are running on the machine. This, of course, lists only instances that are running. To get a list of all instances running on a Unix/Linux system, use a variation of this command:

```
$ ps -ef|grep "ora_.*pmon" | awk '{print $6}' \| awk -F_ '{print $2}'
```

> **Tip ✎**
>
> The output of the `ps` command can obviously be different in different versions of Unix. You may have to change the column that this command prints from `$6` to something else to get this command to work for you.

On Windows, each Oracle SID has its own service named `OracleService` *`SID`*, where *`SID`* is the instance name. You can get a list of running Windows instances using this command:

```
C:> net start | find "OracleService"
   OracleServiceXYZ
   OracleServiceABC
```

If you would like to make this list even better, download the `sed` command from the GnuWin32 project (http://gnuwin32.sourceforge.net), and add it to the list of commands. The following `sed` command tells `sed` to strip off all characters up to the string `OracleService`. This leaves you with a list of instance names.

```
C:> net start | find "OracleService" \
| sed "s/.*OracleService//"
XYZ
ABC
```

> **Tip ✎**
>
> Again, the best thing you can do is pick one of these methods (preferably the *oratab*/ registry method), then publish it and enforce it.

Physical Backups Without rman

Many Oracle environments prefer the supported nature of `rman`. They also enjoy the way that you can completely integrate your commercial backup software with `rman`. It's also the only way to get true incremental backups of your datafiles. In addition, you can also use `rman` to back up to disk without purchasing an agent for your commercial backup system.

However, some DBAs prefer what Oracle calls *user-managed backups*. They put their database into backup mode prior to backing up and take it out of backup mode after backup. Sometimes this is due to a long history and familiarity with user-managed backups; sometimes it is the cost of either the disk you would need to use `rman` without a media manager or the cost of the media manager. For whatever reason, approximately half of Oracle customers perform user-managed backups, but this percentage goes down every day.

This section discusses methods that can be used to safely back up Oracle without using `rman`. You can back up to disk and then back that disk up using your normal backup procedures, or you can back up directly to tape.

Like most RDBMSs, Oracle databases can reside on cooked filesystem files or raw disk devices (raw disks are available only in Unix). Even if raw devices are available, many Oracle DBAs put their databases on cooked files. One of the reasons for this is that if all of the database files are accessible via the filesystem, backing up is very simple. Any standard copy utility (e.g., `cp`, `copy`) or any backup utility (e.g., `dump`, `cpio`, or commercial utility) can copy the data. If you are running Oracle on Unix and decide to put your

Oracle database on raw partitions, you need to use a tool that can back up raw partitions (e.g., `dd`).

Backups can be done offline (a *cold* backup) or online (a *hot* backup). If you're going to perform user-managed backups, you must back up all of the following:

- Datafiles
- Control files
- Online redo logs (if performing a cold backup)
- The parameter file
- Archived redo logs
- Password file if used
- The *ORACLE_HOME* directory

Cold Backup

A cold backup of an Oracle database that is based on filesystem files is the easiest of all database backups because most companies already have some system that backs up their server's filesystems. It could be a homegrown program that runs `dump` or `ntbackup`, or it could be a commercial backup product. To perform a cold backup of an Oracle database, simply perform an orderly shutdown of Oracle (not `shutdown abort`) before running the normal backup. An orderly shutdown performs a checkpoint, flushing any changed data stored in memory to disk, and then stops all processes that allow access to the database. This procedure puts all Oracle files into a clean, consistent, quiescent state.

This procedure assumes, of course, the use of filesystem files. An Oracle database running on a Unix server also could be sitting on raw devices. A cold backup of such a

database requires a little more effort because you need to understand the structure of the database. The procedure starts out the same, by shutting down the database. A filesystem backup at this point, though, gets only the executables and any database objects that reside in the filesystem, such as the control file. The database itself requires extra effort. The first thing to figure out is where all the database files are. A script can “ask” Oracle this question by querying `v$datafile`, but that script would have to be written. Once the locations of all files are known, `dd` can back them up to a file somewhere in the filesystem or send them directly to a backup volume. If they are backed up to a file in the filesystem, it must be done before the normally scheduled filesystem backup. That way, the files are automatically backed up to a backup volume.

Therefore, backing up an Oracle database that uses raw partitions is harder than backing one up that is based on filesystem files. This is one reason why Oracle DBAs have historically used the filesystem to store their database files, even though raw partitions yield slightly better performance.

Hot Backup

If an Oracle database is providing the data for the customer service web page or any other application that requires 24-hour uptime, a cold backup is not acceptable because it requires that the database be shut down on a regular basis. Even if this is done late at night, customers accessing the web page may do so at any time. A company has a much better online image if it is able to leave the web page up all the time. What is needed, then, is a hot, or online backup.

> **Tip ✎**
>
> The database must be running in `archivelog` mode in order to run hot backups.

A hot backup requires quite a bit more work than a cold backup, and this is even more true when the cold backup is of a database using raw devices. The following steps must be taken every time a hot backup is performed:

1. Find out the names and locations of all tablespaces and the datafiles they reside on:
2. a. If running Oracle 10*g*R2 or later, you must ask Oracle for a list of all datafiles, using the SQL command shown here:

   b.
   ```
   SQL> select file_name from sys.dba_data_files;
   /oracle/product/10.2.0/oradata/orcl/users01.dbf
   /oracle/product/10.2.0/oradata/orcl/sysaux01.dbf
   /oracle/product/10.2.0/oradata/orcl/undotbs01.dbf
   /oracle/product/10.2.0/oradata/orcl/system01.dbf
   /oracle/product/10.2.0/oradata/orcl/example01.dbf
   ```

   c. If running a version prior to Oracle 10*g*R2, you must ask Oracle for a list of all datafiles and tablespaces, using the SQL command shown here:

   d.
   ```
   SQL> select tablespace_name, file_name from sys.dba_data_files;
   USERS  /oracle/product/10.2.0/oradata/orcl/users01.dbf
   SYSAUX  /oracle/product/10.2.0/oradata/orcl/sysaux01.dbf
   UNDOTBS1
   /oracle/product/10.2.0/oradata/orcl/undotbs01.dbf
   SYSTEM  /oracle/product/10.2.0/oradata/orcl/system01.dbf
   EXAMPLE
   /oracle/product/10.2.0/oradata/orcl/example01.dbf
   ```

3. Ask Oracle for the location of the archived redo logs using a SQL command like the one shown here. Since you can now specify multiple locations, change the

query to show those parameters that are `like 'log_archive_dest_%'`.

4.
```
SQL> select name,value from v$parameter where name
SQL> like 'log_archive_dest_%';
log_archive_dest_1  location=/backups/archive
```

5. Put the database into backup mode:

6. a. If running Oracle 10*g*R2 or later, you can put the entire database into backup mode using the SQL command `alter database begin backup`.

   b. If running a version prior to 10*g*R2, put each tablespace into backup mode using the SQL command `alter tablespace` *`tablespace_name`* `begin backup`.

> **Tip ✎**
>
> Because more redo is generated when tablespaces are in backup mode, there are performance ramifications to placing the entire database in backup mode. If placing the entire database into backup mode at once creates too much of a load on your server, place a few tablespaces into backup mode at a time and back up just their datafiles. This is obviously a lot more complicated than the method proposed here.

1. Copy each tablespace's datafiles to an alternate location such as disk or tape, or run your commercial backup program to back up all filesystems (and possibly raw devices) to disk or tape.

2. Take the database out of backup mode:

3. a. If running Oracle 10*g*R2 or later, you can take the entire database out of backup mode by running the

SQL command `alter database end backup`.

b. If running a version prior to 10*g*R2, take each tablespace out of backup mode using the SQL command `alter tablespace` *`tablespace_name`* `end backup`.

4. Switch the redo logfile and make sure it is archived. The best method to do both is to run the SQL command `alter system archive log current`. This switches the logfile but does not return the prompt until the previous redo log has been archived. You can run `alter system switch logfile`, but then you won't be sure that the latest redo log has been archived before you move on to the next step.
5. Back up the control file using the SQL commands `alter database backup controlfile to` *`filename`* and `alter database backup controlfile to trace`.

> **Tip ✎**
>
> Make sure you use both methods to back up the control file; either one may come in handy at different times.

1. Ensure that all archived redo logs that span the time of the backup are backed up.

This is obviously a lot of work. A good knowledge of scripting is required, as well as a good knowledge of the commands necessary to accomplish these tasks. To explain the whole process, let's break it down by section:

Determine structure

Steps 1 and 2 have to do with figuring out where everything is. Make sure you do these steps every single

time you do a backup, not just once or only when you change the structure of the database. This ill-advised method results in a static script that backs up only the tablespaces that were discovered the last time these steps were done. This is too open to human error and not recommended. It is much better to automate these steps and do them each time an instance is backed up. Doing them each time ensures that the configuration data is always current—and that the backups never miss a thing.

Put files into backup mode, and then copy them

Copying the database files while the database is running can be done only by placing one or more tablespaces (or the whole database) in backup mode. The datafiles that are in those tablespaces or that database then can be copied or backed up at will. Since the files are still being written to as you are backing them up, Oracle refers to this as an *inconsistent backup*. Some also refer to this type of backup as a "fuzzy" backup; they are the same thing. The point is that the file is still changing while you are backing it up. Don't worry; Oracle will be able to resolve any inconsistencies in the file during recovery.

Tip ✎

Please see the following section for a refutation of the common misconception that datafiles are not being written to when they are in hot-backup mode.

Back up related files

The datafiles are just one of the many sections of Oracle that need to be backed up. Also back up the control file, the archived redo logs, the configuration files, the

password file if you're using it, the *ORACLE_HOME* directory, and any directories. (If it is a cold backup, you should also copy the online redo logs because they create a complete copy of the database at that point in time.)

Debunking Hot-Backup Myths

What happens during a hot backup is widely misunderstood. I debunk two main myths here:

- Datafiles don't change during hot backup.
- Oracle logs a full copy of every block every time it's changed during hot backup.

Many people believe that while a tablespace is in backup mode, the datafiles within that tablespace are not written to. They believe that all changes to these files are kept in the redo logs until the tablespace is taken out of backup mode, at which point all changes are applied to the datafiles just as they are during a media recovery. (The concept of media recovery is described in the section "Recovering Oracle" later in this chapter.) Although this explanation is easier to understand (and swallow) than how things really work, it is *absolutely not* how hot backups work in Oracle.

Many people believe that when a tablespace is in backup mode, Oracle switches from logging the redo vector to logging full blocks. Neither is correct. It continues to log redo vectors, but it also logs the full image of any block that is changed—but only the first time it changes that block.

oraback Script

Unix Backup & Recovery had a Unix-only script that supported hot and cold backups for Oracle databases using the `begin backup` and `end backup` commands. The script still exists and is available on http://www.backupcentral.com but is not covered in detail in this chapter for two reasons. The first reason is that the script is in a constant state of change, and I didn't want the book to be out of date with the script. (We're currently looking for volunteers to help create a Perl version that will also support Windows. Feel free to join the team.) The second reason is space. This edition of the book got really large, and we had to cut where we could.

A common reaction to these statements is a very loud "What?" followed by crossed arms and a really stern look. (I reacted the same way the first time I heard it.) "How could I safely back up these files if they are changing as I'm backing them up? What do you mean it logs the full image only the first time it changes the block?" Don't worry; Oracle has it all under control. Remember that every Oracle datafile has an SCN that is changed every time an update is made to the file. Also remember that every time Oracle makes a change to a datafile, it records the vector of that change in the redo log. When a tablespace is put into backup mode, the following three things happen:

1. Oracle checkpoints the tablespace, flushing all changes from shared memory to disk.
2. The SCN markers for each datafile in that tablespace are "frozen" at their current values. Even though further updates will be sent to the datafiles, the SCN markers will not be updated until the tablespace is taken out of backup mode.
3. In addition to recording how it changed a particular block (referred to as the *redo vector*), it logs the entire

image of each block the first time it changes it on disk.

After this happens, your backup program works happily through this datafile, backing it up block by block. Since the file is being updated as you are reading it, it may read blocks just before they're changed, after they're changed, or even while they're changing! Suppose that your filesystem block size is 4 KB, and Oracle's block size is 8 KB. Your backup program will be reading in increments of 4 KB. It could back up the first 4 KB of an 8 KB Oracle data block before a change is made to that block, then back up the last 4 KB of that file after a change has been made. This results in what Oracle calls a *split block*. However, when your backup program reaches the point of the datafile that contains the SCN, it backs up the SCN the way it looked when the backup began because the SCN is frozen. Once you take the tablespace out of backup mode, the SCN marker is advanced to the current value, and Oracle switches back to logging change vectors and doesn't worry about full images of changed blocks.

How does Oracle straighten this out during media recovery? It's actually very simple. You use your backup program to restore the datafile. When you attempt to start the instance, Oracle looks at the datafile and sees an old SCN value. Actually, it sees the value that the SCN marker had *before* the hot backup began. When you enter `recover datafile`, it begins to apply redo against this datafile. Since the redo logs contain one complete image of every block that changed during your backup, it can rebuild this file to a consistent state, regardless of when you backed up a particular block of data. The first thing it does is overwrite any blocks for which it contains full images (that is, those blocks that changed during backup). It then applies regular redo against the file to apply any changes that occurred during or after the backup. This is why it

needs only the first image of any block that was changed during the backup. It just needs one image to ensure that the block is put into a known state (as opposed to a split block). It can then apply redo against that block to redo any other changes.

If you're like me, you won't believe this the first time that you read it. So I'll prove it to you. Let's create a table called `tapes` in the tablespace `test`, insert the value "DLT" into it, and force a checkpoint:

```
SQL> create table tapes (name varchar2(32)) tablespace test;
Table created.
SQL> insert into tapes values ('DLT');
1 row created
SQL> commit;
Commit complete.
SQL> alter system checkpoint;
System altered.
```

Now we ask Oracle what block number contains the new value:

```
SQL> select dbms_rowid.rowid_block_number(rowid) blk, name from tapes;
  BLK NAME
------- ----------------
   3 DLT
```

The value "DLT" is recorded in the third data block. Allowing nine blocks for the datafile headers, we can read the third block of data with `dd` and run `strings` on it to actually see that the value is there:

```
$ dd if=/db/Oracle/a/oradata/crash/test01.dbf ibs=8192 skip=11 count=1|strings
1+0 records in
16+0 records out
DLT
```

Place the tablespace in hot-backup mode:

```
SQL> alter tablespace test begin backup ;
Tablespace altered.
```

Now update the table, commit the update, and force a global checkpoint on the database:

```
SQL> update tapes set name = 'AIT';
1 row updated
SQL> commit;
Commit complete.
SQL> alter system checkpoint;
System altered.
```

Extract the same block of data to show that the new value was actually written to disk:

```
$ dd if=/db/Oracle/a/oradata/crash/test01.dbf ibs=8192 skip=11 count=1|strings
1+0 records in
16+0 records out
DLT,
AIT
```

Now we can take the tablespace out of backup mode:

```
SQL> alter tablespace test end backup;
```

This test proves that datafiles are indeed being written to during hot backups!

Physical Backups with rman

`rman` offers a number of advantages over user-managed backups. The following is a quick list of them:

Full support from Oracle

> While user-managed backups are supported by Oracle, you'll find that Oracle's interest goes up even more if you're recovering your database using its favorite tool.

Automatic creation of multiple simultaneous backup streams

> If you've got a big database, this will come in very handy. All you have to do is specify how many simultaneous backups you need, and `rman` will generate them.

Incremental backups for all versions since 8*i*

> `rman` performs a true incremental backup, sending only those blocks that have changed since the last backup.

Low-impact incremental backups for all versions since 10g

> With 10*g*, incremental backups got much better. Prior to 10*g*, it took a lot of I/O to determine which blocks needed to be on the incremental backup. This saved a lot of space and bandwidth, but definitely didn't save I/O. 10*g* stores a record of changed blocks in a simple bitmap that's consulted by `rman` during incremental backups, making them much faster and creating much less work for the server.

Use of the flash recovery and flashback features

> Flash recovery and flashback are integrated into `rman`, and you get to use these wonderful features if you use `rman`.

Backup optimization

Since `rman` is keeping track of all backups, you have the option of using backup optimization, which doesn't back up parts of the database it knows it already has a backup of, such as read-only tablespaces.

Restore optimization

Instead of having to figure out which datafiles need to be restored and selecting them from your backup software, you can simply tell `rman` to `restore database`. It then automatically determines which datafiles are in need of a restore and restores them.

During media recovery, `rman` comes in quite handy as well. Instead of having to figure out where your archived redo logs are, and restoring all of them to disk for the restore to work, `rman` completely automates retrieval of redo logs for media recovery. It can also be told to keep a certain amount on disk, automatically deleting the oldest versions from disk to bring in newer versions from tape.

If you use `rman` with a commercial backup utility, such as those covered in Chapter 8, it can actually deliver a completely integrated backup solution for all Oracle databases.

Tip ✎

Since this book is focused on free and open-source backup methods, this chapter does not cover how to integrate `rman` with a commercial backup utility.

In addition to these `rman` features, a fully commercial solution would add these features:

Full support from your commercial backup product vendor

> Don’t expect your backup vendor to know anything about user-managed backups.

Direct backup to tape or virtual tape or other disk on your backup server (instead of being forced to back up to a filesystem accessible to the Oracle server)

> If you’re using `rman` without a media manager, you’ll need to back up to a filesystem accessible on the database server. This can present a number of challenges to some environments. A media manager gives you the most flexibility when choosing backup targets.

Usage of full capabilities of backup product vendor

> Commercial backup products have come a long way in the past few years and include a number of features that we discussed in Chapter 8. You get access to all these features for your `rman` backups as well.

A complete inventory of your backups, allowing a “point-and-click” restore

> You can continue to restore things from the `rman` interface, or you can use your backup software’s ability to perform the restore via its interface.

> **Tip ✎**
>
> If you're interested in these features but have been intimidated by the pricing of enterprise-level backup software products, you may consider Oracle's Secure Backup product. It's got a very simple pricing structure. Oracle doesn't charge per client, per server, or per gigabyte; it charges only a certain amount per backup device.

Once a third-party backup software product, your `rman` script, or Oracle's Secure Backup initiates an `rman` backup, `rman` then does all the internal communication that it needs to do to supply the backup utility with *n* threads of data. `rman` records the time of the backup for future reference and passes it to the backup product if there is one. After things have been set up, it is also possible for a DBA to run the `rman` command from the command line. This command then calls the appropriate programs to connect with the backup utility or back up directly to disk. If sent to a backup utility, it responds to this as to any other backup request, loading volumes as necessary, and accepting the backup from `rman`.

`rman` is supported in Oracle 8*i* and following. You can specify to restore a tablespace, database, datafile, or even a block within a datafile. It knows what files comprise a tablespace or database, and then automatically selects and restores the most recent backup of those files. This really takes the guesswork out of the DBA's hands. The program knows what needs to be restored, it knows where the backups are, and it automatically goes through the process of restoring the appropriate items and can automatically apply media recovery against them as well. This is far better than having to do all this work yourself.

rman is too complex to be covered in detail in a chapter of this size; consult Oracle's documentation for an explanation of how rman works. What we can look at is a list of major features that have been added to rman since it was introduced in 8*i*, resulting in a much more powerful and usable backup command. I'll then provide some general guidance on the use of rman.

Important New rman Features

The following section lists important features of rman that have been introduced since it was originally released. Next to each feature is listed the version of Oracle that introduced it.

Crosscheck and delete functionality (8*i*)

: Prior to these commands, the rman catalog could get out of sync with the media manager catalog. These commands automatically synchronize the two, including deleting rman catalog entries for backups that have been expired by the media manager.

`create`, `upgrade`, and `drop catalog` commands (8*i*)

: These commands automate a number of important catalog operations, replacing a number of scripts and other commands in the process.

Automatic backup name generation (8*i*)

: You no longer have to specify the format parameter with a backup. rman automatically creates a unique name for each piece.

Control file autobackups (9*i*)

: If `configure controlfile autobackup` is set to `on`, rman automatically backs up the control file and server

parameter file any time there is a structural change to the database or any time you issue a backup command.

Backup of server parameter files (9*i*)

Server parameter files are automatically backed up any time you issue the backup command, allowing you to issue the `restore spfile` command (if you enable the `controlfile autobackup` feature).

Backup of archived logs that need backups; deletion of those that don't (9*i*)

Previous methods of specifying which archived logs to back up were a bit clunky and required you to figure out which ones to back up. Now you can specify `not backed up` *n* `times`. Files that have not been backed up the specified number of times are backed up. You can also specify that files that have been backed up the appropriate number of times should be deleted.

Persistent rman configurations (9*i*)

Previous to this, you had to specify a number of arguments to `rman` every time you ran a backup. Now you can create persistent values for things like automatic channels, channel parallelism, retention policies, and backup options.

Block media recovery (9*i*)

Block media recovery can perform media recovery on individual blocks in an individual datafile while the datafile remains online.

Flash recovery area (10*g*)

You can tell `rman` to use a single location on disk for the location of all backup and recovery files. This area is called the flash recovery area.

Optimization of incremental backups (10*g*)

Previous versions of rman required a number of reads of an Oracle database to determine which blocks should be included in an incremental backup, resulting in an I/O load that was actually higher than a full backup. As of 10*g*, Oracle keeps track of which blocks have changed in each datafile. (The DBA must enable this feature.) Now rman can simply ask for changed blocks, significantly reducing the I/O load of an incremental.

Incrementally updating backups (10*g*)

rman can apply the changes found in an incremental backup to a previous backup of a datafile, resulting in a new full backup of that file, without having to actually back up the entire file.

Backup set compression (10*g*)

Oracle can compress backups to better use disk space, reducing the space or bandwidth required by 50 to 75 percent. This is not the default option, but you can make it your default option using a persistent rman parameter.

Recovery through resetlogs (10*g*)

Prior to this feature, you were forced to take a backup after opening the database with the resetlogs option because you couldn't use backups prior to that time. This is no longer the case.

Global stored scripts (10*g*)

rman scripts can be stored in the rman catalog that can be accessed by any target database.

Automating rman

The following section contains some tips on automating rman backups, especially those in large environments.

> **Warning ☠**
>
> This section is absolutely not a replacement for the `rman` documentation. We cannot cover in a few pages what they cover in an entire book.

Like all other Oracle commands, `rman` assumes you know the name of the instance that you're backing up. Therefore, you must first obtain a list of Oracle instances.

Create a recovery catalog

Yes, you can use `rman` without a recovery catalog. In fact, looking at the release notes shows that Oracle continues to enhance what you can do without a catalog. If you don't use a catalog, `rman` data is stored in the control file. Storing your recovery information inside the thing you want to recover is never a good idea. It would be much better to have a centralized recovery catalog that contains all the recovery information for all instances. It's also a good idea if this catalog is on a different server than any of the databases it is tracking.

It's also much easier to create the catalog now by just using the `create catalog` command in `rman`. Large environments might want to use more than one `rman` catalog and back up each of those catalogs using an `rman` instance that's connected to the other catalog. You might also consider performing a hot or cold backup of the `rman` catalog by some method that doesn't use `rman`.

Warning ☠

Always keep a record of your DBID. This may come in very handy if you lose your control file, especially if you have multiple databases with the same name.

Create persistent parameters

Each persistent parameter that you create is one less parameter that you have to specify on the command line. The ultimate goal here is that you would simply be able to tell `rman` to `backup database`, and everything else would come from these parameters. The parameters that can be set persistently can be viewed by using the `show all` command in an `rman` session. It will list all of the parameters and show you which of them you have customized, and which of them are still set to the default values. Here is the example shown in Oracle's documentation:

```
RMAN> show all;

RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
```

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/oracle/product/10.2.0/db_1/dbs/snapcf_orcl.f'; # default
```

Using persistent parameters greatly simplifies your rman scripts. To create or modify these parameters, simply connect to rman and issue commands like those shown here:

```
RMAN> configure backup optimization on;
RMAN> configure device type disk backup type to compressed backupset;
RMAN>configure controlfile autobackup on;
```

Create rman scripts

You can create rman scripts to back up an entire database, a tablespace, a datafile, a control file, or an archive log. These scripts can be stored in the filesystem, or they can be stored in the rman catalog, allowing all instances using that catalog to use the scripts. Once stored there, they can be executed by connecting to the catalog and issuing the rman command run *script_name*.

For space reasons, we'll cover only two very basic rman scripts that perform a full (level 0) and incremental (level 1) backup of the entire instance as well as switching logfiles and backing up any archived redo logs that were created during the backup. A full or level 0 backup backs up every byte of the database, whereas an incremental or level 1 backup backs up only those bytes that have changed since the last backup. This script assumes that you have controlfile_autobackup set to on.

Here is the rman command file used to perform a full backup. It creates a full (level 0) backup of the database to the default device. The plus archivelog option causes it to switch logfiles (using archivelog current) before the backup, back up any archived logs not backed up (if backup optimization is set to on), then back up the database,

followed by another logfile switch and a backup of any redo logs that were created during the backup:

```
connect target userid/passwd@oracle_sid;
connect catalog userid/passwd@rman_sid;
backup incremental level 0 plus archivelog;
```

> **Warning ☠**
>
> Oracle's documentation tells you to enter the `rman` password on the command line. This makes it available to anyone who can enter `ps -ef`. (The following scripts do not do this, but you can see that it was done by manually entering the passwords into the script.) If you follow this method, make sure you make the scripts readable only by Oracle.

Substitute the appropriate values for *`userid`*, *`password`*, and the SIDs for the target and catalog databases. (The target database is the one to be backed up.)

The database.inc.rman command file (level 1 backups)

This is an incremental companion to the full script shown previously. It performs an incremental backup of those bytes that have changed in the database since the level 0 backup was taken:

```
connect target userid /passwd@oracle_sid;
connect catalog userid /passwd@rman_sid;
backup incremental level 1 plus archivelog;
```

Tip ✎

Long-time users of `rman` should note the missing `allocate channel` commands and the lack of an `alter system archive log current` command, and backup of the archive logs at the end. The former is handled with the persistent parameters, and the latter is handled by the `plus archivelog` argument.

Flashback

Oracle 10*g* introduced a wonderful set of features collectively referred to as flashback. Some use the flash recovery area that was discussed in the architecture section, and others use the new recycle bin or the undo tablespace. Where physical backups (the focus of this chapter) are mainly designed to recover from equipment and disk failures, flashback offers a number of interesting ways to recover from human error. Here is a quick summary of these features. Be sure to consult your documentation on how to use flashback.

Flashback query

: This allows a user to query data at a previous point in time in order to reconstruct data that was lost or deleted by accident. The data needed for this feature is stored in the undo tablespace.

Flashback version query

: Allows a DBA to view changes to the rows of a table over time. The data needed for this feature is stored in the undo tablespace.

Flashback transaction query

: Allows a DBA to view changes to the database at the transactional level. The data needed for this feature is stored in the undo tablespace.

Flashback database

: This is a CDP-style recovery of Oracle. It allows you to simply roll Oracle back in time to just before a major logical corruption error happened, such as the drop of a table. The data needed for this feature is stored in the flash recovery area.

Flashback table

: Allows the DBA to recover a table to an earlier point in time. The data needed for this feature is stored in the undo tablespace.

Flashback drop

: Provides “undrop” protection when dropping tables in Oracle. The data needed for this feature is stored in the Oracle recycle bin.

Other Commercial Backup Methods

There is at least one commercial backup tool that backs up Oracle databases without using `rman`. It uses the same tools your scripts would use (e.g., `begin backup`/`end backup`), so it suffers many of the same limitations that such scripts would have. One such product is BMC’s SQL Backtrack. Originally designed for Sybase, this product has been ported to Oracle, Informix, and SQL Server. At this writing, SQL Backtrack does not do volume management. It can, however, interface with some commercial storage managers, allowing them to provide volume management.

Managing the Archived Redo Logs

How common is the question, “Should I have archiving turned on?” Yes, yes, a thousand times *yes*! When in doubt, archive it out! Here’s what is possible only if archiving is enabled:

- Recovery up to the point of failure
- Recovery from a backup that is a month or more old—if all the archived redo logs since then are available
- Performance of a complete backup of the database without even shutting it down

The existence of archive logs does all this without adding significant overhead to the entire process. The only difference between having archiving on or off is whether or not Oracle copies the current redo log out to disk when it “switches” from one redo log to the next. That’s because even with archiving off, it still logs every transaction in the online redo logs. This means that the only overhead associated with archiving is the overhead associated with copying the online file to the archive location, which is why there may be only a 1 to 3 percent performance hit in an environment with many transactions—if there is one at all. Feel free to experiment, but it is very difficult to justify turning off archiving on any production database.

> **Warning ☠**
>
> Remember that the archived redo log destination must not fill up. This causes the production database to stop operating.

In my opinion, there are only two environments in which turning off archiving is acceptable. The first is an environment in which the data does not matter. What type of environment would that be? The only one is a true test environment that is using fake data or data restored from production volumes. No structure changes are being made to this database, and any changes made to the data are discarded. This database does not need archiving and probably doesn't even need to be backed up at all. (Did I just write that?) It should be mentioned, though, that if you're benchmarking a database that will go into production, backup and archiving should be running because testing the application without archiving running does not give realistic results. Tests with archiving running are more realistic—even if all the archive logs are deleted as soon as they are made.

Development databases do not fall into this category because, although the data in a development database may be unimportant, the structure of the database often is highly important. If archiving is off, a DBA cannot restore any development work that he has done since the last backup. That creates the opportunity to lose hours or even days worth of work, just so a development database can be 1 to 3 percent faster. That is a big risk for such a small gain.

The second type of database that doesn't need archive logs is a completely read-only database or a "partially read-only" database where an archive log restore would be slower than a reload of the original data. The emergence of the data warehouse has created this scenario. There are now some databases that have completely read-only tablespaces and never have data loaded into them. This type of database can be backed up once and then left alone until it changes again.

A partially read-only database is one that stays read only for long periods of time and is updated by a batch process that runs nightly, weekly, or even as needed. The idea is that, instead of saving hundreds of redo logs, the database would be restored from a backup that was taken before the load. The DBA then could redo the load. There are two choices in this scenario. The first is to turn off archiving, making sure that there is a good cold backup after each database load. If the load aborted or a disk crashed after the load but before the next backup, you can simply load the older backup and then redo the load. The cold backup costs some downtime, but having archiving off speeds up the loads somewhat. The other option would be to turn on archiving. That allows taking a hot backup anytime and creates the option of using the redo logs to reload the data instead of doing an actual data reload. This method allows for greater backup flexibility. However, depending on the database and the type of data, an archive log restore could take longer than a reload of the original data—especially if it is a multithreaded load. It is a tradeoff of performance for recoverability. Test both ways to see which one works best for you.

Recovering Oracle

Since an Oracle database consists of several interrelated parts, recovering such a database is done through a process of elimination. Identify which pieces work, then recover the pieces that don't work. The following recovery guide follows that logic and works regardless of the chosen backup method. It consists of a number of related steps that can be followed in order. If you're performing user-managed backups, you'll have to perform much of the logic yourself. If you're using `rman`, you'll be able to skip right through a lot of these steps.

Archiving Saves the Day

I know of one company that had a 250 GB database and did not use archiving at all (250 GB was enormous in those days!). The biggest downside to this was that they could not do hot backups, and a cold backup took too long. The result was that they didn't do any backups! The DBAs didn't want to turn on archiving because they said that it would make the batch loads take too long. They also believed that having archiving turned on would somehow cause database corruption. This is just not possible. Again, the only difference between running and not running archiving is whether the old redo log is copied to the archive destination. The rest of the database works exactly the same.

I tried to convince them to turn on archiving. I even bet them that turning on archiving would not add more than a 3 percent overhead to their load times. In other words, a five-hour load would take only 5 hours and 9 minutes. I lost the bet because it took 5 hours and 10 minutes. The DBAs agreed to turn on archiving, and the database received its first backup ever in five years. *Two weeks later*

that database lost five disks—believe it or not. We were able to recover the database overnight with no downtime to the users.

Using This Recovery Guide

The process presented in the following sections for recovering an Oracle database assumes nothing. Specifically, it does not assume that you know the cause of the database failure. By following these steps you'll work through a series of tasks that determine which part(s) of the database are no longer functional. You can then bring the database up as soon as possible while allowing recovery of the pieces that are damaged. ("Damaged" may mean that a file is either missing or corrupted.)

Start with Step 1. If it succeeds, it directs you to Step 10. If the "startup mount" fails, it directs you to Step 2. Each of the steps follows a similar pattern, directing you to the appropriate step following the failure or success of the current step. Assume that prior to any SQL commands, we issued the following commands:

```
$ sqlplus /nolog
SQL>connect /as sysdba;
```

Prior to running any `rman` commands, we connected to the appropriate target database and recovery catalog (if we're using one). Consult your manual for the dozens of different ways you can do that. Our examples didn't use a recovery catalog, and we were logging into the target database using OS-level authentication, so we simply had to run the following command:

```
$ rman target /
```

For each `sqlplus` or `rman` command, I describe the state the database needs to be in prior to running the command.

For example, the paragraph will say something like, “issue the following SQL command against a *mounted, closed database*.”

For more detailed information about individual steps, please consult Oracle’s documentation, especially the *Backup and Recovery Basics* documentation.

These steps can be used on a Windows machine as well as a Unix machine. Differences are noted when necessary.

> **Warning ☠**
>
> Before you begin any recovery, it’s wise to start a support ticket with Oracle. They can be very helpful if things get difficult.

Seriously, Think About rman

You can see that the restore and recovery of an Oracle database can be a very complicated process. If you’re not yet using `rman`, now is the time to think about it. As long as you still have your control files, this 26-step recovery process would be reduced to the following 6 `rman` commands:

```
connect target......
connect catalog.....
startup mount;
restore database;
recover database;alter database open;
```

`rman` automatically figures out what’s broken, restores it, then applies media recovery against it.

Step 1: Try Startup Mount

The first step in verifying the condition of an Oracle database is attempting to mount it. This works because mounting a database (without opening it) reads the control files but does not open the datafiles. If the control files are multiplexed/mirrored, Oracle attempts to open each of the control files listed in the parameter file. If any of them are damaged, the mount fails.

To mount an unmounted database, simply run `sqlplus /nolog`, connect to the database, and enter `startup mount`:

```
SQL> startup mount;
```

If it succeeds, the output looks something like this:

```
SQL> startup mount;
ORACLE instance started.
Total System Global Area       5130648 bytes
Fixed Size                     44924 bytes
Variable Size                  4151836 bytes
Database Buffers               409600 bytes
Redo Buffers                   524288 bytes
Database mounted.
```

If the attempt to mount the database fails, the output looks something like this:

```
SQL> startup mount;
Total System Global Area    5130648 bytes
Fixed Size                  44924 bytes
Variable Size               4151836 bytes
Database Buffers            409600 bytes
Redo Buffers                524288 bytes
ORACLE instance started.
ORA-00205: error in identifying controlfile, check alert log for
more info
```

Tip ✎

If the attempt to mount the database succeeds, proceed to Step 10. If it database fails, proceed to Step 2.

Can You Run sqlplus Commands Against a Down Database?

These steps often require running several different `sqlplus` commands to find out the status of certain parts of the database. Some commands require that the database be mounted while others don't. When the database is not open, queries can be run only against fixed tables, or views. Examples of such views are: *V$DATAFILE*, *V$LOGFILE*, *V$TABLESPACE*, and *V$LOG*.

For a complete list of fixed tables that are available when the database is mounted, run the following command on any Oracle database, even if it's not open:

```
SQL> select * from V$FIXED_TABLE ;
```

Step 2: Are All Control Files Missing?

Don't panic if the attempt to mount the database fails. Control files are easily restored if they were multiplexed/mirrored and can even be rebuilt from scratch if necessary. The first important piece of information is that one or more control file is missing.

Unfortunately, since Oracle aborts the mount at the first failure it encounters, it could be missing one, two, or all of the control files, but so far you know only about the first missing file. So, before embarking on a course of action, determine the severity of the problem by doing a little research.

First, determine the names of all of the control files. You can do that by querying `v$parameter`, which is a fixed table available from an unmounted (or mounted) database.

```
SQL> select value from v$parameter where name like 'control_files'
/db/Oracle/a/oradata/crash/control01.ctl,
/db/Oracle/b/oradata/crash/control02.ctl,
/db/Oracle/c/oradata/crash/control03.ctl
```

Or on Windows, it might look like this:

```
SQL> select value from v$parameter where name like 'control_files'
D:\ORACLE\ORADATA\CRASH\CONTROL01.CTL,
E:\ ORACLE\ORADATA\CRASH\CONTROL02.CTL,
F:\ ORACLE\ORADATA\CRASH\CONTROL03.CTL
```

It's also important to get the name of the control file that Oracle is complaining about. Find this by looking for the phrase `controlfile:` in the alert log. (The alert log can be found in the location specified by the `background_dump_dest` value also available in `v$parameter`.) Use the technique shown previously to get the locations of the control files to get this value. (Typically, the alert log is in the *<ORACLE_BASE>/admin/<ORACLE_SID>/bdump* directory.) In that directory, there should be a file called *alert_<ORACLE_SID>.log*. In that file, there should be an error that looks something like this:

```
Sun Jul 23 18:53:19 PDT 2006 alter database mount exclusive
Sun Jul 23 18:53:19 PDT 2006 ORA-00202: controlfile:
'/db/a/oradata/crash/control01.ctl'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
```

> **Warning ☠**
>
> Some of the following procedures may say to override a potentially corrupted control file. Since you never know which file may be needed, always make backup copies of all the control files before doing any of this. This procedure offers an "undo" option that isn't possible otherwise. (Also make copies of the online redo logs as well.)

With the names of all of the control files and the name of the damaged file, it's easy to determine the severity of the problem. Do this by listing each of the control files and

comparing their size and modification time. (Remember the game “One of these is not like the others” on *Sesame Street*?) The following scenarios assume that the control files were multiplexed/mirrored to three locations, which is a very common practice. The possible scenarios are:

The damaged file is missing or corrupted, and at least one other file is present

> If the file that Oracle is complaining about is just missing, that’s an easy thing to fix. If this is the case, proceed to Step 3.

All control files are missing

> If all the control files are corrupt or missing, they must be rebuilt or the entire database must be restored. Hopefully your backup system has been running the `backup controlfile to trace` command on a regular basis. (The output of this command is a SQL script that rebuilds the control files automatically.) Proceed to Steps 4 through 7.

Step 3: Replace Missing Control File

> **Tip ✎**
>
> If you don’t have the output of a backup control file to trace, you can still build a recreate control file script using a process documented by Oracle. Support will be very helpful here.

Please note the careful choice of words here. We are not going to *restore* the control file, which would imply that we’re going to get it back from backups. Remember that restoring the control file forces us to perform an `alter`

`database open resetlogs`, which is not desirable if we can afford it. We are going to *replace* it by copying one of the other copies that we multiplexed/mirrored it to. Let's see if we can do that before we actually restore one from backup.

> **Tip ✎**
>
> This step applies to both user-managed backups and `rman` backups. It's always better to repair the control file from a multiplexed/mirrored copy than to restore it from backup.

If you're able to identify that at least one of the multiplexed/mirrored copies of the control file is good, then this part is easy. Simply copy another one of the multiplexed/mirrored copies of the control file to the damaged control file's name and location. (The details of this procedure follow.) Once this is done, you can attempt to mount the database again.

> **Warning ☠**
>
> Be sure to make backup copies of *all* of the control files before trying this procedure!

The first thing to do is to get the name of the damaged control file. Again, this is relatively easy. Look in the alert log for a section like this one:

```
Sun Jul 23 18:53:19 PDT 2006 alter database mount exclusive
Sun Jul 23 18:53:19 PDT 2006 ORA-00202: controlfile:
'/db/a/oradata/crash/control01.ctl'
ORA-27037: unable to obtain file status
SVR4 Error: 2: No such file or directory
```

The next step would be to copy a known good control file to the damaged control file's location.

“But I don’t have a good control file!”

It’s possible that there is no known copy of the current control file. If this is the case and you weren’t using your control file to store `rman` backup history, it’s probably best to attempt to use the `create controlfile` script. If you have `rman` backup history in your control file, it’s probably best to actually restore your control file from backup.

To use the `create controlfile` script, proceed to Steps 4 through 7. To restore your control file from backup, proceed to Step 8.

If that’s not possible or probable, try the following procedure. First, make backups of all the control files. Then, one at a time, try copying every version of each control file to all the other locations—excluding the one that Oracle has already complained about because it’s obviously damaged.

> **Tip ✎**
>
> Each time a new control file is copied to multiple locations, return to Step 1.

For example, assume there are three control files: */a/control1.ctl*, */b/control2.ctl*, and */c/control3.ctl*. The alert log says that the */c/control3.ctl* is damaged, and because */a/control1.ctl* and */b/control2.ctl* have different modification times, there’s no way to know which one is good. Try the following steps:

First, make backup copies of all the files:

```
$ cp /a/control1.ctl /a/control1.ctl.sav
$ cp /b/control2.ctl /b/control2.ctl.sav
$cp /c/control3.ctl /c/control3.ctl.sav
```

Or, for a Windows system, run the following commands:

```
C:\ copy C:\CONTROL01.CTL C:\CONTROL01.SAV
C:\ copy D:\CONTROL02.CTL D:\CONTROL02.SAV
C:\copy E:\CONTROL03.CTL E:\CONTROL03.SAV
```

Second, try copying one file to all locations. No sense in copying from *control3.ctl* because it's obviously damaged. Try starting with *control1.ctl*:

```
$ cp /a/control1.ctl /b/control2.ctl
$cp /a/control1.ctl /c/control3.ctl
```

Or, for a Windows system, run the following commands:

```
C:\ copy C:\CONTROL01.CTL D:\CONTROL02.CTL
C:\copy C:\CONTROL01.CTL E:\CONTROL03.CTL
```

Now attempt a startup mount:

```
SQL> startup mount
Sun Jul 23 18:53:19 PDT 2006 alter database mount exclusive
Sun Jul 23 18:53:19 PDT 2006 ORA-00202: controlfile:
'/a/control3.ctl'
ORA-27037: unable to obtain file status
```

This error says that the file that was copied to all locations is also damaged. Now try the second file, *control2.ctl*. This time we have to copy from backup since we overwrote the original with the last step.

```
$ cp /b/control2.ctl.sav /a/control1.ctl
$cp /b/control2.ctl.sav /a/control3.ctl
```

Or, for a Windows system, run the following commands:

```
C:\ copy D:\CONTROL02.SAV C:\CONTROL01.CTL
C:\copy D:\CONTROL02.SAV E:\CONTROL03.CTL
```

Now attempt to do a startup mount:

```
SQL> startup mount;
ORACLE instance started.
Total System Global Area          5130648 bytes
Fixed Size                      44924 bytes
Variable Size                   4151836 bytes
Database Buffers                  409600 bytes
Redo Buffers                    524288 bytes
Database mounted.
```

It appears that *control2.ctl* was a good copy of the control file.

> **Tip ✎**
>
> If you were able to perform the startup mount successfully, proceed to Step 10. If you were unable to identify a good copy of the control file, proceed to Step 4.

Step 4: Are All Datafiles and Redo Logs OK?

> **Tip ✎**
>
> Steps 4 through 7 should only be performed after unsuccessfully performing Steps 1 through 3. The point of these steps are to get ready to rebuild the control file using the control file script.
>
> Steps 4 and 5 are required only prior to performing Step 6.

In 10*g*, Oracle enhanced the `backup controlfile to trace` output to include both a `noresetlogs` and a `resetlogs` option in the output. If you want to use the `noresetlogs` option, all datafiles and online redo logs must be intact. The datafiles can be older versions that were restored from backup because they will be rolled forward by the media recovery. The online redo logs must be current and intact for the `noresetlogs` version of the `create controlfile` script to work. If any online redo log group is completely damaged, you will have to use the `resetlogs` option.

Why? Because the rebuild process uses the online redo logs to identify the current SCN, then identifies it as the current

SCN in the control file. It's OK if one or more of the datafiles have older SCNs. But if a datafile shows that it has an SCN that is more recent than the available online redo logs, it indicates that we're not dealing with the current online redo logs—the control file rebuild process aborts.

If it's likely that one or more of the datafiles or online redo logs is damaged, go to Step 5. If it's more likely that they are all intact, go to Step 6.

Multiplex Control Files and Redo Logs!

Losing all members of an unarchived active log group is one scenario under which data loss is almost assured, and recovering from a lost control file is a real pain as well. Therefore, protect against these scenarios at all costs. Make sure the control files and online redo logs are multiplexed/mirrored!

Making it happen is easy. First, determine where the multiplexed/mirrored redo logs are going to reside. Remember that if the redo log is multiplexed/mirrored three times, it means Oracle has to write every change to all three logs. That means that all three mirrors/multiplexed copies should be on the fastest disks available. Make sure that the different mirrors/multiplexed copies also are located on different disks!

For this example, assume that there are three log groups with one member each. Here is the output of a `select group#, member from v$logfile`:

```
  1 /logs1/redolog01.log
  2 /logs1/redolog02.log
  3 /logs1/redolog03.log
```

For this example, we multiplex/mirror these three logs to */logs2* and */logs3*. I prefer to keep the filenames of the members of a log group the same. Therefore, in this

example, *redolog01.log* is multiplexed/mirrored to */logs1*, */logs2*, and */logs3*. To do this, we issue the following commands:

```
SQL> alter database add logfile member
     '/logs2/redolog01.log' to group 1;
Selection processed
SQL> alter database add logfile member
     '/logs3/redolog01.log' to group 1;
Selection processed
SQL> alter database add logfile member
     '/logs2/redolog02.log' to group 2;
Selection processed
SQL> alter database add logfile member
     '/logs3/redolog02.log' to group 2;
Selection processed
SQL> alter database add logfile member
     '/logs2/redolog03.log' to group 3;
Selection processed
SQL> alter database add logfile member
     '/logs3/redolog03.log' to group 3;
Selection processed
```

These commands create three multiplexed copies/mirrors for each log group. This significantly decreases the chance that all members of a single log group could be damaged.

Step 5: Restore Damaged Datafiles or Redo Logs

If one or more of the datafiles or online redo logs are definitely damaged, follow all the instructions given here to see if there are any other damaged files. (A little extra effort now saves a lot of frustration later.) If it’s possible that all the datafiles and online redo logs are OK, another option is to skip this step and try to recreate the control file now. (An unsuccessful attempt at this does not cause any harm.) If it fails, return to this step. If there is plenty of

time, go ahead and perform this step first. To try to recreate the control files now, proceed to Step 6.

The first thing to find out is where all the datafiles and redo logs are. To determine this, run the following commands on the mounted, closed database.

```
SQL> select name from v$datafile;
/db/Oracle/a/oradata/crash/system01.dbf
/db/Oracle/a/oradata/crash/rbs01.dbf
/db/Oracle/a/oradata/crash/temp01.dbf
/db/Oracle/a/oradata/crash/tools01.dbf
/db/Oracle/a/oradata/crash/users01.dbf
/db/Oracle/a/oradata/crash/test01.dbf
SQL> select group#, member from v$logfile;
1 /db/Oracle/a/oradata/crash/redocrash01.log
3 /db/Oracle/c/oradata/crash/redocrash03.log
2 /db/Oracle/b/oradata/crash/redocrash02.log
1 /db/Oracle/b/oradata/crash/redocrash01.log
2 /db/Oracle/a/oradata/crash/redocrash03.log
3 /db/Oracle/c/oradata/crash/redocrash02.log
```

The Windows output of these commands would look very similar, with different paths, of course.

Let’s check each of the files shown by the preceding command. First, look at the datafiles. Most of the datafiles probably have the same modification time, or there might be a group of them with one modification time and another group with a different modification time. If there are read-only tablespaces, there may be a few datafiles with modification times a lot older than the others. That’s OK. The main thing to look for is a missing file or a zero-length file. Something else to look for is one or more files that have a modification time that is newer than the newest online redo logfile. If a datafile meets any one of these conditions, it must be restored from backup.

Redo logfiles, however, are a little different. Each redo logfile within a log group should have the same modification time. For example, the output of the preceding example command shows that

/db/Oracle/a/oradata/crash/redocrash01.log and */db/Oracle/b/oradata/crash/redocrash01.log* are in log group 1. They should have the same modification time and size. The same should be true for groups 2 and 3. There are a couple of possible scenarios:

One or more log groups has at least one good and one damaged log.

This is why redo logs are multiplexed/mirrored! Copy the good redo log to the location of the damaged redo log. For example, if */db/Oracle/b/oradata/crash/redocrash01.log* is missing, but */db/Oracle/a/oradata/crash/redocrash01.log* is intact, issue the following command:

```
$ cp /db/Oracle/a/oradata/crash/redocrash01.log \
 /db/Oracle/b/oradata/crash/redocrash01.log.
```

Or, if the errors showed that *D:\ORACLE\ORADATA\CRASH\REDOCRASH01.LOG* was OK, but *E:\ORACLE\ORADATA\CRASH\REDOCRASH01.LOG* was missing or corrupt:

```
C:\ COPY D:\ORACLE\ORADATA\CRASH\REDOCRASH01.LOG \
 E:\ORACLE\ORADATA\CRASH\REDOCRASH01.LOG.
```

All redo logs in at least one log group are damaged.

This is a bad place to be. The `create controlfile` script in Step 6 requires that all online redo logs be present to use the `noresetlogs` option. If even one log group is completely damaged, you will have to use the `resetlogs` portion of the script.

If all datafiles are available, but all the control files are missing, proceed to Step 6.

If the database will not open for some other reason, proceed to Step 10.

Step 6: Is There a “Backup to Trace” of the Control File?

> **Tip ✎**
>
> Steps 4 and 5 must be completed prior to this step.

You’ll need the output from the `sqlplus` command `alter database backup controlfile to trace`. It creates a trace file that contains two `create controlfile` scripts. This command should be run (via `cron` on a Unix/Linux system, a Windows scheduled task, or via the DBMS scheduler) on a regular basis. To find out if there is such a script available, follow these instructions. The first thing to find out is the destination of the trace files. This is specified by the `user_dump_dest` value in the `v$parameter` table. (Typically, `user_dump_dest` is set to *`$ORACLE_BASE`* */admin/* *`$ORACLE_SID`* */udump*.) First `cd` to that directory, then use `grep` on Unix or `find` on Windows to look for the phrase `CREATE CONTROLFILE`, as shown in Example 16-1.

Example 16-1. Locating the most recent create controlfile script

```
SQL> select value from v$parameter where name like 'user_dump_dest';
/db/Oracle/admin/crash/udump
$ cd /db/Oracle/admin/crash/udump ; grep 'CREATE CONTROLFILE' * \
 |awk -F: '{print $1}'|xargs ls -ltr
-rw-r-----  1 Oracle  dba     3399 Oct 26 11:25 crash_ora_617.trc
-rw-r-----  1 Oracle  dba     3399 Oct 26 11:25 crash_ora_617.trc
-rw-r-----  1 Oracle  dba     1179 Oct 26 11:29 crash_ora_661.trc
```

On Windows, try these commands:

```
D:\ cd D:\Oracle\admin\crash\udump
D:\type *.TRC|find 'CREATE CONTROLFILE'
```

In Example 16-1, *crash_ora_661.trc* is the most recent file to contain the `create controlfile` script.

If there is a `create controlfile` script, proceed to Step 7. If there is not a `create controlfile` script, and all the control files are missing, proceed to Step 8.

Step 7: Run the create controlfile Script

First, find the trace file that contains the script. The instructions on how to do that are in Step 6. Once you find it, copy it to another filename, such as *rebuild.sql*. If your online redo logs are intact, you can use the `noresetlogs` section; if any online redo log group is completely damaged, you'll need to use the `resetlogs` section. Take a look at the output, and select the appropriate section, deleting all other comments before and after the section.

The file then should look something like the one in Example 16-2.

Example 16-2. Example create controlfile script

```
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS ARCHIVELOG
  MAXLOGFILES 16
  MAXLOGMEMBERS 3
  MAXDATAFILES 100
  MAXINSTANCES 8
  MAXLOGHISTORY 292
LOGFILE
 GROUP 1 '/oracle/product/10.2.0/oradata/orcl/redo01.log' SIZE
50M,
 GROUP 2 '/oracle/product/10.2.0/oradata/orcl/redo02.log' SIZE
50M,
 GROUP 3 '/oracle/product/10.2.0/oradata/orcl/redo03.log' SIZE
50M
DATAFILE
 '/oracle/product/10.2.0/oradata/orcl/system01.dbf',
 '/oracle/product/10.2.0/oradata/orcl/undotbs01.dbf',
 '/oracle/product/10.2.0/oradata/orcl/sysaux01.dbf',
 '/oracle/product/10.2.0/oradata/orcl/users01.dbf',
 '/oracle/product/10.2.0/oradata/orcl/example01.dbf'
CHARACTER SET WE8ISO8859P1
;
```

```
RECOVER DATABASE
ALTER SYSTEM ARCHIVE LOG ALL;
ALTER DATABASE OPEN;
ALTER TABLESPACE TEMP ADD TEMPFILE
'/oracle/product/10.2.0/oradata/orcl/temp01.dbf'
   SIZE 20971520 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;
```

Once the file looks like Example 16-2, add the following line just above the `STARTUP NOMOUNT` line:

```
CONNECT /AS SYSDBA;
```

After you add this line, run the following command on the idle instance, substituting *`rebuild.sql`* with the appropriate name:

```
$ sqlplus /nolog < rebuild.sql
```

This should work without intervention and completely rebuild the control files. You then open the database when it's done.

> **Tip ✎**
>
> If this worked, and you used the `resetlog` option, it would be best to back up the database as soon as possible.
>
> If this didn't work, you can try Step 8. Most likely you will end up at Steps 21 and 23, though.

Step 8: Restore Control Files and Prepare the Database for Recovery

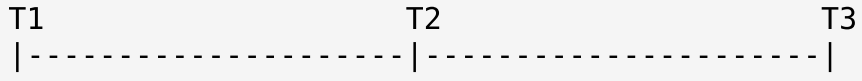
Tip ✎

This step is required only if Steps 2 through 7 have failed.

If the precautions mentioned elsewhere in this chapter were followed, there is really only one scenario that would result in this position—loss of the entire system due to a cataclysmic event. Loss of a disk drive (or even multiple disk drives) is easily handled if the control files are multiplexed/mirrored. Even if all control files are lost, they can be rebuilt using the trace file created by running the `backup controlfile to trace` command.

Stay Away from This Step!

Hopefully, this section is for learning purposes only, because it's not a situation you want to be in. Recovering from the loss of all control files (without the use of the `create controlfile` script) requires opening the database with the `resetlogs` option. When forced to do this, there is one big ramification if you're running a version prior to 10*g*: Oracle cannot use redo logs to roll through this action. Consider this drawing:



Suppose that a backup was made at time T1 and an `open database resetlogs` performed at time T2. Also suppose that a backup was not taken immediately after the recovery, and it is now time T3. You might think that you could take the backup from time T1 and use the redo logs to roll forward to time T3, but that is not possible if an `alter database open resetlogs` was performed at time T2. That is why you *must* perform an immediate backup after opening the database with the `resetlogs` option.

> All the more reason you should upgrade to a recent version!

Follow the next steps, starting with restoring the control files from backup. Chances are that the database files need to be restored as well. This is because it is difficult to use a control file that is older than the most recent database file. (Oracle complains and aborts if this happens.) To find out if the control file is newer than the datafiles, try the following steps without overwriting the database files and see what happens.

1) Restore control files from backup

The very first step in this process is to find and restore the most recent backup of the control file.

If you're using `rman` with a catalog and you're backing up the control file explicitly, you can use the `restore controlfile` command:

```
RMAN> restore controlfile
```

If you've set `controlfile autobackup` to `on`, connect to `rman`, and issue the `restore controlfile from autobackup` command:

```
RMAN> restore controlfile from autobackup
```

If you're performing user-managed backups, you need to issue the `backup controlfile to` *`filename`* command in `sqlplus`. Locate the backed up control file, and copy it to all of the locations and filenames shown when running a `select value from v$parameter where name like "control_files"`.

Again, this backup control file should be more recent than the most recent database file in the instance. If this isn't the case, Oracle will complain.

2) Start up mount

To find out if the control file is valid and has been copied to all the correct locations, attempt to start up the database with the `mount` option. (This is the same command from Step 1.) If you're running `rman`, you can simply issue this command after your `restore controlfile` command:

```
RMAN> startup mount
```

If you're performing user-managed backups, run the following command on the mounted, closed database:

```
SQL> startup mount;
```

Once this step has been completed, proceed to Step 9.

Step 9: Recover the Database

> **Tip ✎**
>
> This step is required only if Steps 2 through 7 have failed.

Once the control file is restored with a backup copy, attempt to recover the database using the backup control file.

Recover and open the database with rman

If you're using `rman`, this section is easy. Just issue `recover database` as your next command. This single command handles recovering the database even if you're using a backup control file.

```
RMAN> recover database;
```

Here's an area where rman really shines. If recovery requires applying archived redo logfiles that have been backed up, rman automatically restores them from the appropriate location, applies them, then deletes them. The entire process occurs automatically, regardless of where the archived redo logs have to come from.

If you did not use a backup control file, you just need to open the database:

```
RMAN> alter database open;
```

If you did use a backup control file, you need to open the database with the resetlogs option:

```
RMAN> alter database open resetlogs;
```

Attempt to recover database manually

Since recovering the database with a backup control file requires the alter database open resetlogs option, it never hurts to try recovering the database normally first:

```
SQL> recover database;
```

If the backup control file option is required, Oracle will complain:

```
ORA-00283: Recover session cancelled due to errors
...
ORA-01207: file is more recent than controlfile - old controlfile
```

> **Tip ✎**
>
> If the recover database command works, proceed to Step 10. If it doesn't, attempt to recover the database using the backup control file, as described next.

If Oracle complains, you need to recover the database using the backup controlfile option. Attempt to recover

the database using the following command on the mounted, closed database:

```
SQL> recover database using backup controlfile
```

If it works, the output will look something like Example 16-3.

Example 16-3. Sample output of recover database command

```
ORA-00279: change 38666 generated at 03/14/06 21:19:05 needed for
thread 1
ORA-00289: suggestion :
/db/Oracle/admin/crash/arch/arch.log1_494.dbf
ORA-00280: change 38666 for thread 1 is in sequence #494
```

Tip ✎

If Oracle complains, there are probably some missing or corrupted datafiles. If there are, return to Steps 4 and 5. Once any missing or corrupted datafiles are restored, return to this step and attempt to recover the database again.

Sometimes you can be trapped in a Catch-22 when you're recovering databases and Oracle complains about datafiles being newer than the control file. The only way to get around this is to use a backup version of the datafiles that is older than the backup version of the control file. If this is the case, proceed to Steps 21 and 23. Media recovery rolls forward any changes that this older file is missing.

Oracle requests all archived redo logs since the time of the oldest restored datafile. For example, if the backup that was used to restore the datafiles was from three days ago, Oracle needs all archived redo logs created since then. Also, the first logfile that it asks for is the oldest logfile that it wants.

If you're performing user-managed backups, and you need older archived redo logs, you have to restore them yourself. If you have to find and apply the logs yourself, the most efficient way to roll through the archived redo logs is to have all of them sitting uncompressed in the directory that Oracle suggests as the location of the first file. (This, of course, requires much more storage than the `rman` method.) If this is the case, simply enter `auto` at the prompt. Otherwise, specify alternate locations, or press enter as it asks for each one, giving time to compress or remove the files that it no longer needs.

If it is able to do so, Oracle automatically rolls through all the archived redo logs and the online redo log. It then says `Media recovery complete`.

However, once Oracle rolls through all the archived redo logs, it may prompt for the online redo log. It does this by prompting for an archived redo log with a sequence number that is higher than the most recent archived redo log available. This means that it is looking for the online redo log. Try answering its prompt with the names of the online redo logfiles that you have. Unfortunately, as soon as you give it a name it doesn't like, it makes you start the `recover database using backup controlfile` command again.

For example, suppose that you have the following three online redo logs:

```
/oracle/data/redolog01.dbf
/oracle/data/redolog02.dbf
/oracle/data/redolog03.dbf
```

When you are prompting for an archived redo log that has a higher sequence number than the highest numbered archived redo log that you have, answer the prompt with one of these files (e.g., */oracle/data/redolog01.dbf*). If the file

that you give it does not contain the sequence number it is looking for, you will see a message like the following:

```
ORA-00310: archived log contains sequence 2; sequence 3 required
ORA-00334: archive log: '/oracle/data/redolog01.dbf'
```

Oracle cancels the database recovery, requiring you to start it over. Once you get to the same prompt again, respond with a different filename, such as */oracle/data/redolog02.dbf*. If it contains the recovery thread it is looking for, it responds with a message like the following:

```
Log applied.
Media recovery complete.
```

If, after trying all the online redo logs, Oracle still asks for a log that you do not have, simply enter `cancel`.

Alter database open resetlogs

Once the media recovery is complete, the next step is to open the database. As mentioned earlier, when recovering the database using a backup control file, it must be opened with the `resetlogs` option. Do this by entering the following SQL command against the mounted, closed database:

```
SQL> alter database open resetlogs;
```

Later versions of Oracle allow you to create a control file with a `resetlogs` option. If this was done, you can simply open the database and do not need the `resetlogs` option here.

If you're running a version of Oracle prior to 10*g*, you *must* take a backup immediately after recovering the database with the `resetlogs` option! Even if you're running 10*g*, it would be good to take a backup as soon as you can. While 10*g* databases can recover through a `resetlogs` incident, you will have to redo the restore and recovery you just did.

It is best if it is a cold backup after shutting down the database, but a hot backup would be better than nothing. Just remember that if you allow the database to become operational, and something happens before you get it backed up again, there is a risk that:

- The entire recovery might need to be performed again.
- All changes made after using the `resetlogs` option are lost, if running a version prior to 10*g*.

> **Tip ✎**
>
> If the database did not open successfully, return to Step 1 and start over.
>
> If the database did open successfully, perform a backup of the entire database immediately—preferably a cold one. Congratulations! You're done!

Step 10: Does "alter database open" Work?

> **Tip ✎**
>
> If you shut down the database prior to this step, you'll need to run `startup mount` again.

If the `startup mount` worked, this is actually only the second step that you perform. Mounting the database checks only the presence and consistency of the control files. If that works, opening the database is the next step. Doing so checks the presence and consistency of all datafiles, online redo logfiles, and any rollback segments or

undo segments. To open the database, run the following command on the mounted, closed database:

```
SQL> alter database open;
```

If the attempt to open the database worked, Oracle simply says `Database Altered`. If this is the first attempt to open the database, and no datafiles or rollback segments were taken offline, you're done!

> **Tip ✎**
>
> If directed to this step by Step 23 (damaged log groups) and the attempt to open the database failed, return to Step 21 to recover the entire database.
>
> If the database did open, proceed to Step 15.

Big shortcut for rman users

If you're using `rman`, and the attempt to open the database did not work, you have a big choice to make. You can continue to follow this procedure to determine what's wrong and fix things one at a time, or you can issue two commands and be done. If the latter idea sounds better, try these two commands:

```
RMAN> restore database ;
RMAN>recover database ;
```

User-managed backups: Read on

If you're performing user-managed backups, or you'd like to perform the rest of these steps manually with `rman`, you now need to find out why the database wouldn't open. The output varies depending on the condition. Here is a listing

of what some of those conditions may be, accompanied by what the error might look like when that condition occurs:

Missing datafile

```
ORA-01157: cannot identify datafile 1 - file not found
ORA-01110: datafile 1:
'/db/Oracle/a/oradata/crash/system01.dbf'
```

Corrupted datafile

A corrupted datafile can generate a number of different errors. For instance, it may mimic a missing datafile:

```
ORA-01157: cannot identify datafile 1 - file not found
ORA-01110: datafile 1:
'/db/Oracle/a/oradata/crash/system01.dbf'
```

It also may completely confuse Oracle:

```
ORA-00600: internal error code, arguments: [kfhcfh1_01], [0],
[], [], [],
```

A corrupted datafile also may cause a “failed verification check” error:

```
ORA-01122: database file 1 failed verification check
ORA-01110: datafile 1:
'/db/Oracle/a/oradata/crash/system01.dbf'
ORA-01200: actual file size of 1279 is smaller than correct
size of 40960 blocks
```

These are just a few examples of the types of errors that Oracle may give if a datafile is corrupted.

Missing member of any online log group

If the redo logs are multiplexed/mirrored and one or more of the multiplexed/mirrored copies is lost but at least one good copy of each online redo log remains, Oracle opens the database without any errors displayed to the terminal. I’m sure they consider this a feature, but it sure would be nice if it would at least complain. It’s great that it opens the database. I just wish that this potentially dangerous situation were a little more

visible. The only error is a message like the following one in the alert log:

```
Errors in file
/db/Oracle/admin/crash/bdump/crash_lgwr_10302.trc:
ORA-00313: open failed for members of log group 2 of thread 1
```

All members of any online log group are corrupted

However, if all members of any online log group are corrupted, Oracle *will* complain, and the database will not open. The error might look something like this:

```
ORA-00327: log 2 of thread 1, physical size less than needed
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

Missing all members of any online log group

A similar problem occurs if all members of an online log group are missing. Oracle will complain, and the database will not open. The error looks something like this:

```
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

Damaged undo tablespace

```
ORA-01122: database file 1 failed verification check
ORA-01110: datafile 1:
'/oracle/product/10.2.0/oradata/orcl/undotbs01.dbf'
ORA-01200: actual file size of 1279 is smaller than correct
size of 40960 blocks
```

Damaged rollback segment

If a rollback segment is damaged, the error looks like the following:

```
ORA-01545: rollback segment 'USERS_RS' specified not
available

Cannot open database if all rollback segments are not
available.
```

Damaged datafile

A damaged datafile is actually very easy to recover from. This is a good thing because this occurs more often than any other problem. Remember that there is only one copy of each datafile, unlike online redo logs and control files that can be multiplexed/mirrored. So, statistically speaking, it's easier to lose one datafile than to lose all multiplexed/mirrored copies of a log group or all multiplexed/mirrored copies of the control file.

Oracle also can recover parts of the database while other parts of the database are brought online. Unfortunately, this helps only if a partially functioning database is of any use to the users in your environment. Therefore, a database that is completely worthless unless all tables are available will not benefit from the partial online restore feature. However, if the users can use one part of the database while the damaged files are being recovered, this feature may help to save face by allowing at least partial functionality during an outage. (Because doing an online (partial) restore actually makes more work for you, you should seriously investigate whether this option will help if you use it—before you actually have to use it. That way, when you need to do a large restore, that question will already be answered.)

There are three types of datafiles as far as recovery is concerned:

- The first type of datafile in a recovery is a datafile that is not required to start the database. Historically, this

was the *system* tablespace. In current versions, required tablespaces include the *system*, *sysaux*, and *undo* tablespaces. If a datafile is part of a tablespace that is not required to start the database, recovering this file (with the database online or offline) is very easy.

- The second type is a datafile for the temporary tablespace that has been declared a tempfile. Such a restore is easy because Oracle simply recreates the tempfile for you when you open the database.
- The third type of datafile is also a nonsystem datafile but one that happens to contain a rollback segment. Since rollback segments are needed to open the database, recovering such a file with the database online is difficult.

> **Tip ✎**
>
> Again, you should migrate to *undo* tablespaces and undo segments as soon as you can.

The final type of datafile is a file contained within a required tablespace (*system* , *sysaux*, or *undo* tablespaces in 10*g*, and *system* in previous versions). This datafile cannot be recovered with the database online because the database cannot be brought online without it.

Damaged log group

If all members of a log group are damaged, there is great potential for data loss. The entire database may have to be restored, depending on the status of the log group that was damaged and the results of some attempts at fixing it. This may seem like a broken record, but this is why multiplexing/mirroring the log groups is so important.

If the error refers to a damaged log group, one option is to proceed directly to Step 17. However, to verify that nothing else is wrong, you should probably read the rest of this step and proceed to the next one.

Damaged required tablespace

If any datafiles belonging to required tablespaces are damaged, then you have to restore and recover them offline. The database cannot be opened without them.

If the error refers to a required tablespace, one option is to proceed directly to Step 11. However, to verify that nothing else is wrong, you should probably read the rest of this step and proceed to the next one.

Damaged rollback segment

> **Tip ✎**
>
> If you're still using rollback segments, you need to look at this part of the step. If you're using automatic undo management, it does not apply to you.

Since Oracle has to open the datafiles that contain this rollback segment before it can verify that the rollback segment is available, this error will not occur unless a datafile has been taken offline. If Oracle encounters a damaged datafile (whether or not it contains a rollback segment), it will complain about that datafile and abort the attempt to open the database.

Remember that a rollback segment is a special part of a tablespace that stores rollback information. Rollback information is needed in order to undo (or roll back) an uncommitted transaction. Since a crashed database almost

always contains uncommitted transactions, recovering a database with a damaged rollback segment is a little tricky. As previously mentioned, a damaged datafile may be taken offline, but Oracle will not open the database without the rollback segment.

If the error indicates that there is a damaged rollback segment, proceed to Step 18.

Before going any farther...

Remember that Oracle will stop attempting to open the database as soon as it encounters an error with one file. This means, of course, that there could be other damaged files. If there is at least one damaged datafile, now is a good time to see whether other files are damaged. Detailed instructions on how to do that are provided in Step 5.

Once you know the names of all the damaged files, you can recover them as described next.

How media recovery works

If any datafiles are restored from backup, the `sqlplus` or `rman` `recover` command is needed. These commands use the archived and online redo logs to redo any transactions that have occurred since the time that the backup of a datafile was taken. You can recover a complete database, a tablespace, or a datafile by issuing the commands `recover database`, `recover tablespace` *`tablespace_name`*, or `recover datafile` *`data_file_name`*, respectively.

With `rman` or user-managed backups, you can simply issue the command `recover database`. Both automatically figure out what needs to have media recovery applied to it and perform the appropriate recovery. The difference between `rman` and user-managed backups comes when you need an

archived log that is available only on backup. If `rman` needs to obtain any archived redo logs from backup, it automates restoring them and deleting them as necessary. If you're performing user-managed backups, you need to perform this step yourself.

You can also apply media recovery to an individual datafile by issuing the SQL commands below against a mounted, closed database:

```
SQL> recover datafile '/db/Oracle/a/oradata/crash/datafile01.dbf'
```

This command allows the restore of an older version of a datafile and uses redo to roll it forward to the point of failure. For example, if you took a backup of a datafile on Wednesday night, and that datafile was damaged on Thursday evening, you would restore that datafile from Wednesday night's backup. Of course many transactions would have occurred since Wednesday night, making changes to the datafiles that you restored. Running the command `recover` [*`database`*|*`tablespace`*|*`datafile`*] reapplies those transactions to the restored datafile, rolling them forward to Thursday evening.

This recovery can work in a number of ways. After receiving the `recover` command, Oracle prompts for the name and location of the first archived redo log that it needs. If that log and all logs that have been made since that log are online, uncompressed, and in their original location, enter the word `auto`. This tells Oracle to assume that all files that it needs are online. It therefore can automatically roll through each log.

In order to do this, all files that Oracle will need must be online. First, get the name of the oldest file, because that is the first file it will need. That filename is displayed immediately after issuing the `recover` command:

```
ORA-00279: change 18499 generated at 02/21/06 11:49:56 needed for
thread 1
ORA-00289: suggestion :
/db/Oracle/admin/crash/arch/arch.log1_481.dbf
ORA-00280: change 18499 for thread 1 is in sequence #481
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
```

In the preceding example, the first file that Oracle needs is */db/Oracle/admin/crash/arch/arch.log1_481.dbf*. Make sure that this file is online and not compressed or deleted. If it is deleted, restore it from backup. If it is compressed, uncompress it and any archived redo logfiles in that directory that are newer than it, because Oracle may need all of them to complete the media recovery. If you're not using `rman`, it might be necessary to delete some of the older archived redo logs to make enough room for the files that need to be uncompressed. Once all archived redo logs that are newer than the one requested by Oracle have been restored and uncompressed, enter `auto` at the `Specify log` prompt. `rman` can be configured to restore only one redo log at a time from tape, apply that redo log, then delete it, removing the space issue. It can also be configured to keep a certain amount on disk using the `maxsize` parameter.

> **Warning ☠**
>
> This whole part of the restore will obviously work only if you are running in `archivelog` mode. Make sure you're using that option wherever you can.

If there isn't enough space for all of the archived redo logs to be uncompressed, a little creativity may be required. Uncompress as many as possible, and then press Enter each time it suggests the next file. (Pressing Enter tells Oracle that the file that it is suggesting is available. If it finds that it is *not* available, it prompts for the same file again.) Once it has finished with one archive log, compress

that log, and uncompress a newer log, because it will be needed shortly. (Obviously, a second window is required, and a third window wouldn't hurt!)

At some point, it may ask for an archived redo log that is not available. This could mean that some of the archived redo logs or online redo logs are damaged. If the file cannot be located or restored, enter `cancel`.

More details on media recovery are available in Oracle's documentation.

If any of the damaged datafiles is a member of any required tablespaces, proceed to Step 12. If none of them is a member of any required tablespaces, proceed to Step 13.

Step 11: Are There Damaged Datafiles for Required Tablespaces?

If the damaged file is part of a required tablespace (*system*, *sysaux*, *undo*, and *temp* in 10*g*, or *system* in previous versions), an offline recovery is required. Unfortunately, Oracle complains only that the datafile is missing—without saying what *kind* of datafile it is. Fortunately, even if Oracle is down, there is an easy way to determine which files belong to the required tablespaces. Finding out if the datafile contains a rollback segment is a little more difficult, but it is still possible. If you use `rman` with a recovery catalog, you can connect to the catalog, set the DBID of the database, and issue the `report schema` command. This command displays the files, locations, and whether they contain rollback segments.

```
RMAN> report schema;
Report of database schema
File Size(MB) Tablespace RB segs Datafile Name
---- ---------- ---------------- ------- ------------------
1  307200 SYSTEM   NO   /oracle/oradata/trgt/system01.dbf
```

```
2  20480  UNDOTBS  YES  /oracle/oradata/trgt/undotbs01.dbf
3  10240  CWMLITE  NO   /oracle/oradata/trgt/cwmlite01.dbf
4  10240  DRSYS    NO   /oracle/oradata/trgt/drsys01.dbf
5  10240  EXAMPLE  NO   /oracle/oradata/trgt/example01.dbf
6  10240  INDX     NO   /oracle/oradata/trgt/indx01.dbf
7  10240  TOOLS    NO   /oracle/oradata/trgt/tools01.dbf
8  10240  USERS    NO   /oracle/oradata/trgt/users01.dbf
```

To find out which datafiles are in the *system* tablespace, you'll need to query `sys.dba_data_files`:

```
SQL> select file_name, tablespace_name from sys.dba_data_files;
/oracle/oradata/trgt/system01.dbf SYSTEM
/oracle/oradata/trgt/undotbs01.dbf UNDOTBS
/oracle/oradata/trgt/cwmlite01.dbf CWMLITE
/oracle/oradata/trgt/drsys01.dbf DRSYS
/oracle/oradata/trgt/example01.dbf EXAMPLE
/oracle/oradata/trgt/indx01.dbf INDX
/oracle/oradata/trgt/tools01.dbf TOOLS
/oracle/oradata/trgt/users01.dbf USERS
```

This example report shows three datafiles that are members of the *system* , *sysaux*, and *undo* tablespaces. In your configuration, however, there may be multiple datafiles assigned to these tablespaces.

If any of the damaged datafiles is a member of a required tablespace, proceed to Step 12. If none of them are members of required tablespaces, proceed to Step 13.

Step 12: Restore All Datafiles in Required Tablespaces

Unlike other tablespaces, required tablespaces (*system*, *sysaux*, *undo*, and *temp* in 10*g*, or *system* in previous versions) must be available in order to open the database. Therefore, if any members of these tablespaces are damaged, they must be restored now. Before doing this, make sure that the database is not open. (It is OK if it is mounted.) To make sure, run the following command on the mounted, closed database:

```
SQL> select status from v$instance;
STATUS
-------
MOUNTED
1 row selected.
```

The preceding example shows that this instance is mounted, not open. If the database is not open, restore the damaged files from the most recent backup available. If you're performing user-managed backups, you'll need to figure out where those backups are and restore them. If you're using `rman`, simply issue a single command:

```
RMAN> restore database;
```

`rman` automatically figures out which files need to be restored and restores only those files from the most recent backup. It does not restore files that appear to be alright.

Once all damaged files in the required tablespace are restored, run the following command on the mounted, closed database. If you want to just bring these tablespaces online and then fix other tablespaces later, you'll need to apply media recovery just against the individual tablespaces. This may be more complex and may take more time than just restoring all damaged datafiles and issuing a single `recover database` command.

To recover just the required tablespaces, run individual `recover` commands against them. The following examples show media recovery being applied against the *system* tablespace:

```
RMAN> recover tablespace system;
```

Or in SQL:

```
SQL> recover tablespace system;
```

If you're recovering multiple tablespaces, it's probably best if you just issue the `recover database` command, which applies media recovery against the entire database.

Once this command has completed, the required tablespaces are recovered to the time of failure.

If it does complete successfully, and no other datafiles are damaged, return to Step 10. For more information about the `recover tablespace` command, read the earlier section "How media recovery works" at the end of Step 10. If there are other datafiles to recover, proceed to Step 13.

Step 13: Damaged Nonrequired Datafile?

So far, we have mounted the database, which proves that the control files are okay. It may have taken some effort if one or more of the control files were damaged, but it succeeded. We also have verified that any required tablespaces are intact, even if they required a restore and recovery. Most of the rest of this procedure concentrates on disabling damaged parts of the database so that it may be brought online as soon as possible. The process of elimination identifies all damaged datafiles once the database is opened successfully. They can then be easily restored.

If there are damaged datafiles that are not members of required tablespaces, proceed to Step 14. If there are no more damaged datafiles, proceed to Step 17.

Step 14: Take Damaged Datafile Offline

To open a database with a damaged, nonrequired datafile, take the datafile offline.

> **Warning ☠**
>
> This step should not be used against datafiles that contain rollback segments. Those should be restored prior to bringing the database online.

If this instance is operating in `archivelog` mode, just take the datafile offline. It can be restored and recovered later, after the instance has been brought online. Here's the command to do this:

```
SQL> alter database datafile 'filename' offline;
```

If the instance is operating in `noarchivelog` mode, that's a different problem. Oracle does not allow the datafile to be taken offline because it knows it can't be brought back online without media recovery. Without `archivelog` mode, there is no media recovery. If this is the case, you need to treat any damaged datafiles as required and go back to Step 12.

Once any damaged files are taken offline, return to Step 10 and attempt to open the database again.

Step 15: Were Any Datafiles Taken Offline?

> **Tip ✎**
>
> Perform this step only if the database has been opened.

This step is really a very simple question: if the database opens without taking any datafiles offline, you're done.

If you had to take some datafiles offline to open the database, you need to restore and recover them now. To

find out if any datafiles were taken offline, run the following command:

```
SQL> select name from v$datafile where status = 'OFFLINE' ;
NAME
------------------
/db/oracle/a/oradata/crash/tools01.dbf
/db/oracle/a/oradata/crash/users01.dbf
```

If some datafiles were taken offline to open the database, proceed to Step 16. If you're unsure, proceed to Step 16.

Step 16: Restore and Recover Offline Datafiles

If any datafiles were taken offline, you need to restore them and bring them online.

Restore the damaged datafiles

Once the names of the datafiles that need to be restored are determined, restore them from the latest available backup. If you're performing user-managed backups, you have to decide which files to restore and where to restore them from. If you're using `rman`, you only need to issue the following command:

```
RMAN> restore database ;
```

Once they are restored, recovery within Oracle can be accomplished in three different ways. These ways vary greatly in complexity and flexibility. Examine the following three media recovery methods and choose whichever one is best for you.

Datafile recovery

If there are a small number of datafiles to recover, this may be the easiest option. As each file is restored, issue the `recover datafile` command against it, and then bring it online. The following commands work in `rman` or `sqlplus`:

```
recover datafile 'datafile_name' ;
alter database datafile 'datafile_name' online ;
```

The downside to this method is that media recovery may take a while for each datafile. If recovering multiple datafiles within a single tablespace, this is probably wasting time.

Tablespace recovery

This is the hardest of the methods, but it may work faster than the previous method if there are several damaged datafiles within a tablespace. If forced to leave the partially functional database open while recovering the damaged datafiles, and there are several of them to recover, this is probably the best option.

First, find the names of all datafiles and the tablespace to which they belong. Since the database is now open, this can be done in one step, as demonstrated in Example 16-4.

Example 16-4. Listing dba_data_files

```
SQL> select file_name, tablespace_name from dba_data_files;
Statement processed.
FILE_NAME
 TABLESPACE_NAME
------------------------------------------------------------------------
---------------
 ------------------------------
/db/oracle/a/oradata/crash/users01.dbf
 USERS
/db/oracle/a/oradata/crash/tools01.dbf
 TOOLS
/db/oracle/a/oradata/crash/temp01.dbf
 TEMP
/db/oracle/a/oradata/crash/rbs01.dbf
```

```
 RBS
/db/oracle/a/oradata/crash/system01.dbf
 SYSTEM
/db/oracle/a/oradata/crash/test01.dbf
 TEST
```

Once all of the datafiles are restored and the names of all the tablespaces that contain these datafiles have been determined, issue the `recover tablespace` command against each of those tablespaces. Before doing so, however, each tablespace must be taken offline, as shown in Example 16-5. The commands in Example 16-5 work in `rman` or `sqlplus`.

Example 16-5. Tablespace-based recovery

```
alter tablespace tablespace_name1 offline;
recover tablespace tablespace_name1 ;
alter tablespace tablespace_name1 online;
alter tablespace tablespace_name2 offline;
recover tablespace tablespace_name2 ;alter tablespace tablespace_name2 online;
```

It’s obvious that this method is quite involved! It’s not pretty, and it’s not easy, but it allows recovery of multiple tablespaces while the instance continues to operate. If a partially functioning database is of any value to the users, this method may be their best friend.

Database recovery

This is actually the easiest method, but it requires that the database be shut down to perform it. After restoring all the database files that were taken offline, close the database, and issue the `recover database` command.

Once all the database files are restored, issue the commands shown in Example 16-6. These commands will work in `rman` or `sqlplus`.

Example 16-6. Normal database recovery

```
shutdown immediate ;
startup mount ;
recover database ;alter database open ;
```

To make sure that all tablespaces and datafiles have been returned to their proper status, run the commands shown in Example 16-7.

> **Tip ✎**
>
> Depending on the version of Oracle, some of the following commands may display a status only if there is something "interesting."

Example 16-7. Obtaining the names of all datafiles, control files, and logfiles

```
SQL> select name, status from v$datafile
NAME
 STATUS
-------------------------------------------------------------------
---------------
 -------
/db/oracle/a/oradata/crash/system01.dbf  SYSTEM
/db/oracle/a/oradata/crash/rbs01.dbf     ONLINE
/db/oracle/a/oradata/crash/temp01.dbf    ONLINE
/db/oracle/a/oradata/crash/tools01.dbf   ONLINE
/db/oracle/a/oradata/crash/users01.dbf   ONLINE
/db/oracle/a/oradata/crash/test01.dbf    ONLINE
6 rows selected.
SQL> select member, status from v$logfile
/oracle/product/10.2.0/oradata/orcl/redo03.log
/oracle/product/10.2.0/oradata/orcl/redo02.log
/oracle/product/10.2.0/oradata/orcl/redo01.log
SQL> select name, status from v$controlfile;
/oracle/product/10.2.0/oradata/orcl/control01.ctl
/oracle/product/10.2.0/oradata/orcl/control02.ctl
/oracle/product/10.2.0/oradata/orcl/control03.ctl
```

Example 16-7 shows that all datafiles, control files, and logfiles are in good condition. (In the case of the logfiles and control files, no status is good status.)

> **Tip ✎**
>
> Once any datafiles that were taken offline have been restored and recovered, you're done.

Step 17: Is There a Damaged Online Log Group?

When we refer to a "damaged online log group," we mean that all members of a log group are damaged. If at least one member of a multiplexed/mirrored log group is intact, Oracle opens the database and simply puts an error message in the alert log. However, if all members of a log group are damaged, the database will not open, and the error will look something like this:

```
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

> **Tip ✎**
>
> If there is no error like this, there is no damaged log group. Proceed to Step 18.

The first thing that must be determined is the status of the damaged log group. The three statuses to worry about are `CURRENT`, `ACTIVE`, and `INACTIVE`. To determine the status of the damaged log group, run the following command on the mounted, closed database:

```
SQL> select group#, status from v$log;
```

The output looks something like this:

```
GROUP#   STATUS
---------- ----------------
     1 INACTIVE
     2 CURRENT
     3 ACTIVE
3 rows selected.
```

The preceding example shows that log group 1 is inactive, group 2 is current, and group 3 is active. Here is an explanation of these statuses and how they affect recovery:

Current

The current log group is typically the log Oracle is currently writing to. In the case of a crash, it's the one to which Oracle was writing when the failure occurred. It will be shown as current until the server is brought online and a log switch occurs.

Active

An active log group is usually the log group that Oracle just finished writing to. However, until a checkpoint occurs, this group is still needed for media recovery. Since a log switch always forces a checkpoint, a status of active is actually very rare. In fact, the only way to see this (before the system crashed) is to run the preceding command while a checkpoint is in progress. (In a properly tuned database, this is a very short period of time.)

Inactive

An inactive log group is one that is not being used by Oracle at the moment. It has been archived.

To determine what action to take next, first get the number of the log group whose logfiles are damaged. The preceding example error reads `open failed for members of log group 2`. Reference this number against the log groups listed by the `select * from v$log` command. In

the previous example, log group 2 was `CURRENT` at the time the database crashed.

> **Tip ✎**
>
> If the damaged log group was current, proceed to Step 20. If it was active, proceed to Step 23. If it was inactive, proceed to Step 25.

Step 18: Are Any Rollback Segments Unavailable?

> **Tip ✎**
>
> As mentioned repeatedly, rollback segments are the older way to handle undo. If you have switched to undo segments, this step doesn't apply to you. Undo segments were handled in Step 12.

If a rollback segment is damaged, Oracle will complain when attempting to open the database. The error looks like the following:

```
ORA-01545: rollback segment 'USERS_RS' specified not available
Cannot open database if all rollback segments are not available.
```

If the preceding error is displayed when attempting to open the database, proceed to Step 19. If not, return to Step 10.

Step 19: Recover Tablespace Containing Unavailable Rollback Segment

> **Tip ✎**
>
> Perform this step only if directed to do so by Step 18.

The first thing that must be determined is which tablespace the damaged rollback segment is in. Unfortunately, there is no fixed view that contains this information. That means that it will have to be discovered through common sense and deduction.

> **Tip ✎**
>
> This is why it is very helpful to put the rollback segments in dedicated tablespaces with names that easily identify them, such as RBS1. It's even more helpful if the datafiles are named something helpful as well. For example, create a separate tablespace called RBS1, and call its datafiles *rollback01.dbf*, *rollback02.dbf*, and so on. That way, anyone who lands in this scenario will know exactly which datafiles contain rollback data.

First, remember that this error is not displayed unless a datafile has been taken offline. To get a complete list of files that were taken offline, run the following command on a mounted, closed database:

```
SQL> select TS#, name from v$datafile where status = 'OFFLINE' ;
NAME
-----------------------------------------------------------------
---------------
5 /db/oracle/a/oradata/crash/test01.dbf
1 row selected.
```

Now, find the name of the tablespace that contains this datafile:

```
SQL> select name from v$tablespace where TS# = '5' ;
NAME
```

```
--------------------------------------------------------------------
--------------
TEST
1 row selected.
```

Admittedly, the previous example was easy. There was only one datafile that was offline, which made finding its tablespace pretty easy. What if there were multiple datafiles contained within multiple tablespaces? How do you know which one contains the rollback segment? Unfortunately, there is no way to be sure while the database is closed. The rest of this step is simple. Restore any files that were taken offline, and use either the `recover datafile` or `recover tablespace` commands to roll them forward in time. If there are only one or two damaged datafiles, it's probably quicker to use the `recover datafile` command. If there are several damaged datafiles, especially if they are all in one tablespace, the `recover tablespace` command is probably easiest. Either way will work.

> **Tip ✎**
>
> Once any datafiles that contain rollback segments have been restored and recovered, return to Step 10 and attempt to open the database again.

Step 20: Is the Current Online Log Damaged?

> **Tip ✎**
>
> Perform this step only if instructed to do so by Step 17. Otherwise, return to Step 17 now.

If the current online log group is damaged, you'll see a message like the following when you attempt to open the database:

```
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

In the preceding example, a `select group#, status from v$log` command also would have shown that log group 2 was `CURRENT` at the time of failure.

This is the worst kind of failure to have because there definitely will be data loss. That is because the current online log can be required to restart even a fully functioning database. If the current file contains redo needed to bring the instance online, you will not be able to open the database without a full restore and incomplete recovery.

First, try opening the database with an `alter database open resetlogs`, which induces Oracle to recreate the online redo logs. If that works, you're done.

If you cannot open the database with the `resetlogs` option, the only other step is to restore an older version of the control file. Unfortunately, you can't restore only the control file because the datafiles then would be more recent than the control file. The only remaining option is to restore the entire database.

Tip ✎

If the current online redo log is damaged, proceed to Step 22. If not, proceed to Step 24.

Step 21: Restore and Recover All Database Files from Backup

> **Warning ☠**
>
> There are only two reasons to perform this step. The first is if instructed to do so by Step 20. The other is if there was an unsuccessful attempt to open the database after performing either Steps 25 or 27. This step is the most drastic method of recovery and should not be performed unless absolutely necessary.

Perform this step only after verifying (or rebuilding or restoring) the control files, and verifying that all members of the current online log group are damaged. This procedure is relatively easy. Simply determine the names and locations of all the datafiles, and restore them from their latest backup.

> **Warning ☠**
>
> Restore only the datafiles, not the control files. Do not restore or overwrite the control files unless instructed to do so by Step 9!

Again, if you're using `rman`, you can do all of this with two commands:

```
RMAN> restore database ;
RMAN>recover database ;
```

If you're performing user-managed backup, you need to use your backup system to restore all datafiles. To determine

the names of all the datafiles, run the following command on the mounted, closed database:

```
SQL> select name from v$datafile ;
```

Once all datafiles are restored, you should issue a media recovery against them.

```
SQL> recover database ;
```

Once all datafiles are restored and recovered, proceed to Step 22.

Step 22: Run alter database open resetlogs

> **Warning ☠**
>
> Perform this step only if instructed to do so by Step 21. This is another drastic step that should be performed only if necessary!

The `alter database open resetlogs` command causes Oracle to open the database after clearing all contents of the online redo logfiles. Since there is no way to undo this step, it is a good idea to make copies of the online redo logfiles now. To find out all their names, run the following command on a mounted, closed database:

```
SQL> select member from v$logfile ;
```

To create an "undo" option, copy each file to *<filename>.bak*. After making a backup of the online redo logfiles, run the following command on a mounted, closed database:

```
SQL> alter database open resetlogs ;
Statement processed.
```

If the database opens, congratulations!

Warning ☠

If you're running a version prior to 10*g*, make a backup of this database *immediately*, preferably with the database shut down. That is because prior to 10*g*, Oracle could not roll through this point in time using the redo logs. Oracle versions prior to 10*g* must have a full backup taken after using the `open resetlogs` command in order to apply media recovery to this database using any redo logs that are made after the `open resetlogs` was performed.

It's also a good idea to perform a backup as soon as possible even if you're running 10*g*. A hot backup will suffice.

Once that backup is completed, you're done!

Step 23: Is an Active Online Redo Log Damaged?

Tip ✎

Perform this step only if instructed to do so by Step 17. Otherwise, return to Step 17 now.

If an active online log group is damaged, you'll see a message like the following when attempting to open the database:

```
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

In the preceding example, a `select group#, status from v$log` command also would have shown that log group 2 was active at the time of failure.

Remember that an active log is one that is still needed for recovery. Depending on how the database crashed, there might still be data in the buffers. If you can get that data to flush to disk with a checkpoint, you can turn this active log into an inactive log and delete it.

To perform a checkpoint, proceed to Step 24. If there are no damaged active online redo logs, proceed to Step 25.

Step 24: Perform a Checkpoint

The way to attempt to recover from the scenario in Step 23 is to perform a checkpoint. If it is successful, the database should open successfully. To perform a checkpoint, issue the following command on the mounted, closed database:

```
SQL> alter system checkpoint local ;
Statement processed.
```

Be patient. The reason that there is an active log group is probably that the checkpoint took a long time in the first place, and the database crashed during the checkpoint. Wait for Oracle to say that the checkpoint succeeded or failed. If it succeeded, Oracle will simply say, `Statement processed`. If it failed, there could be any number of Oracle errors.

After issuing the checkpoint, even if it is unsuccessful, return to Step 10, and attempt to open the database. If this attempt fails, return to Step 21 and recover the entire database.

Step 25: Is an Inactive Online Redo Log Damaged?

Tip ✎

Perform this step only if instructed to do so by Step 17. Otherwise, return to Step 17 now.

If an inactive online log group is damaged, you'll see a message like this when attempting to open the database:

```
ORA-00313: open failed for members of log group 2 of thread 1
ORA-00312: online log 2 thread 1:
'/db/Oracle/b/oradata/crash/redocrash02.log'
ORA-00312: online log 2 thread 1:
'/db/Oracle/a/oradata/crash/redocrash03.log'
```

In the preceding example, a `select group#, status from v$log` command also would show that log group 2 was inactive at the time of failure.

By comparison, this one should be a breeze. An inactive log is not needed by Oracle. If the log group is not needed, simply drop it and create another to replace it.

To drop and add an inactive log group, proceed to Step 26.

Step 26: Drop/Add a Damaged, Inactive Log Group

Tip ✎

Perform this step only if instructed to do so by Step 25.

In all the previous examples, the damaged log group was group 2. Before we drop that group, we should make sure that we can add it back easily. Ensure that all the original redo log locations are still valid. To do this, get the names of all the members of that log group:

```
SQL> select member from v$logfile where GROUP# = 2 ;
```

For this example, Oracle returned the values:

```
/logs1/redolog01.dbf
/logs2/redolog01.dbf
/logs3/redolog01.dbf
```

Verify that the locations of all these files are still valid. For this example, assume that */logs3* is completely destroyed, and we are relocating all its contents to */logs4*. Therefore, the members of log group 2 will be */logs1/redolog01.dbf*, */logs2/redolog01.db*, and */logs4/redolog01.dbf*.

To drop log group 2, issue the following command on a mounted, closed database:

```
SQL> alter database drop logfile group 2 ;
```

Once that command completes successfully, add the log group back to the database. To do this, issue the following command (remember that we have replaced */logs3/redolog01.dbf* with */logs4/redolog01.dbf*):

```
SQL> alter database add logfile group 2 ('/logs1/redolog01.dbf', '
/logs2/redolog01.dbf', '/logs4/redolog01.dbf') size 500K ;
Statement processed.
```

Once this command completes successfully, return to Step 10 and attempt to open the database.

You’re Done!

If you’ve made it this far, you’re done! All datafiles, control files, and logfiles should be online. Take a backup of the entire database immediately, preferably a cold one with the database down. If that can’t be done, then perform a hot backup.

Logical Backups

Physical backups protect you from physical damage, such as a damaged disk drive. A logical backup protects you from logical damage, such as when your DBA accidentally deletes an important table. Logical backups are done by Oracle's data pump or export utility, which stores the data in a binary file that is useful only to Oracle.

Data pump and export are really used only for data exchange between databases. It has a number of disadvantages when using it for backups:

No transaction recovery

: It is very important to note that exports are a picture of the database at some point in time, and they can be used to recover that table only to that point in time. There is no way to apply transaction logs (redo logs) on top of an imported table to bring it up to date. So although it might be quicker to restore a table using an import, it might be better just to restore the entire tablespace. The `recover until change` option can be used to redo all transactions except for the one that deleted the table.

Longer than physical backups

: Exports usually take longer than physical backups because there is a lot of checking going on during the process. Depending on the speed of the system, this may be a little bit longer or quite a bit longer!

Full export requires restrict mode or the consistent=y option

: You need to ensure that the export is consistent. One way to do that is to shut down and then open the database in `restrict` mode. That way no one can make

changes to the database while the export is running. (Of course, this means that no one can access it either.) Another way is to add the `consistency=y` option to the export. This causes the export to view the entire database as it existed at the moment the export started.

Performing a Logical Backup

The type of export covered here is an export of the entire database, also known as a *full export*. The commands to do a full export are found in Example 16-8. Substitute the appropriate *`username`*, *`passwd`*, and *`file_name`*. (The username and password need to have the appropriate permissions to do an export.)

Example 16-8. Sample database export

```
$ exp userid=username/passwd full=Y consistent=Y constraints=Y file=file_name
```

This performs a full (every table) export of *`ORACLE_SID`* to *`file_name`*. For more information on the `exp` command, consult the manual.

Recovering with a Logical Backup

If you made any logical backups, or exports, using Oracle's `exp` utility, they may be imported using Oracle's `imp` utility. In order to use the following command, you must substitute the appropriate *`username`*, *`passwd`*, and *`file_name`*. For `level`, you need to use `system` or `restore`. (When to use these levels is covered next.)

```
$ imp username/passwd inctype=level full=Y file=file_name
```

A Broken Record

There are some common threads that appear throughout this chapter. Here they are once again:

Multiplex/mirror the redo logs

: If all members (or the only member) of an active or current log group are lost, there will be data loss. If the redo logs are not multiplexed/mirrored, the chances of this happening are much greater than if the redo logs are multiplexed/mirrored. If the redo logs are multiplexed/mirrored, the chance that all members of a log group would be damaged is incredibly small. A little research and a little effort up front will significantly reduce the chances of data loss.

Watch the alert log

: Even if the redo logs are multiplexed/mirrored, one or more members of a log group may be damaged while the database is operating. The only notification will be an entry in the alert log. Automate the checking of the alert logs for error messages. Otherwise, there may be only one functioning member of a log group, and you may not even know it.

Multiplex/mirror the control files

: A significant portion of this recovery procedure is dedicated to recovering from the loss of all control files. If they are not multiplexed/mirrored, they should be—another example that Prior Proper Planning Purges the Person from the Performance of Painful Procedures.

Use archivelog mode

: Without `archivelog` mode, Steps 21 and 23 could replace the entire recovery procedure. If one datafile is

lost, restore all of them and open the database with the `resetlogs` option. All changes since the last cold backup will be lost, and a cold backup is the only kind of backup possible because hot backups require `archivelog` mode.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 17. Sybase Backup and Recovery

Sybase Adaptive Server Enterprise, or ASE, provides an excellent and easily understood set of server maintenance facilities. Unfortunately, there is no vendor-provided system for server maintenance, backups and restores, and alarming. You need to create your own server maintenance system, either with shell scripts or Perl. Luckily, this is not a complicated task. Your backup system can be scheduled via the Sybase job scheduler or a system scheduler. This chapter guides you through the process of creating your own backup system, using simple examples for illustration.

> **Tip ✎**
>
> This chapter was contributed by Ed Barlow. Ed has been contributing to the Sybase community for over 15 years and offers a number of free tools at his web site http://www.edbarlow.com.

The fundamental structure of this chapter mirrors the other database chapters in this book. The purpose is not to replace Sybase documentation but to cover normal server maintenance, backup, and recovery in a concise and understandable manner. Rarely used and deprecated features are skipped to provide more space for the information you need to understand normal server maintenance. As with the other chapters, I first review the database architecture, then describe common terms. I then cover common utilities, commands, and procedures, continuing with a description of common maintenance, backup, and recovery tasks, including details on how to script them. I finish with a step-by-step disaster recovery procedure.

The Sybase server documentation is excellent. It is split into several "books," each of which documents a particular task. The three books you need to know about are the *Adaptive Server Enterprise Reference Manual: Commands*, which details the syntax of all the Sybase SQL commands; the *System Administration Guide*, which is an excellent guide to server administration; and the *Troubleshooting and Error Messages Guide*, which is a handy reference containing walkthroughs for the errors you might encounter. These manuals are stored at http://sybooks.sybase.com. Checking out the documentation should be your first instinct whenever you see an error message or want information on a task you do not understand. The online guides are well organized and easily searchable. The troubleshooting guide is particularly useful, allowing you to look up error messages and often giving a step-by-step fix. The Sybase documentation is free, easily accessible online, and very well written.

For the purposes of this chapter, the term *maintenance scripts* refers to all scripts that perform necessary dataserver maintenance. This includes database backups, database audits, database consistency checks, object-level backups, and running `update statistics`. I also include information on how to recover from backups.

Sybase Architecture

As in the other database chapters, it is important to understand the design of the database that is being backed up, so this chapter starts with a discussion of Sybase architecture. This is similar to the information provided in Chapter 15, but contains specific information about Sybase. Just as in the overview chapter, we start with the power user's view of the database, continue with a view for the

database administrator, and finish with a diagnostic and recovery procedure.

Overview of the Sybase Architecture

Sybase works similarly on Windows and Unix systems and is shipped in two flavors—client software and server software. Applications use client libraries to communicate with the database server. Client libraries are distributed in a variety of formats and are normally compiled into applications.

Every system that has Sybase applications that run client programs requires a small client software installation. Database servers are normally dedicated systems running only Sybase ASE database software. Sybase server software is high-performance and supports both multiple processors and multithreading. The Sybase system also runs one or more additional server processes such as backup servers (one per system), replication servers, and monitor servers.

Sybase architecture is very similar to that of Microsoft SQL Server because in 1998, Microsoft purchased a license for Sybase's source code and distributed it as Microsoft SQL Server. The architecture, structure, and internal commands are therefore similar between the two products. In Sybase, though, you need to script backups and server maintenance or schedule them using the ASE Job Scheduler. A strength of Sybase is that it scales extremely well from Windows systems through larger Unix systems and has a well-deserved reputation as a high-performance database system.

Sybase Command-Line Utilities

There are a few common Sybase command-line utilities that you should know about. These applications exist both in your `OCS-` *`VERSION`* and `ASE-` *`VERSION`* directory. It does not matter which program you use, but you should make sure the version you use is reasonably current. Very old versions of these clients do not support all the datatypes that the server does.

isql

`isql` is a no-frills command that sends SQL commands to your database server. The basic syntax is `isql –U` *`user`* `-P` *`password`* `-S` *`server`*. `isql` responds with a prompt, (>) at which you can type SQL commands to your server. Transact-SQL (T-SQL) command batches are sent from `isql` to your server whenever it finds the string `go` on a separate line. `isql` can be used to test connectivity and to perform basic server administration.

Other interesting commonly used arguments to `isql` include `–i` *`file`* and `–o` *`file`*, which specify input and output files respectively. It is also common, when scripting SQL commands, to use Unix *here document* syntax. For example:

```
isql –Usa –SSERVERNAME –Ppassword << EOF
exec sp_who
go
exitEOF
```

bcp

The `bcp` command imports and exports a single table or view in or out of the database (to a file). This command is used commonly for data loads and for file extracts. `bcp` runs at the operating system level like `isql` and takes the following parameters:

```
bcp [[database_name.]owner.]table_name {in | out} datafile [-c|-n] -S
servername -U username -P password –t{fld_delimeter} –r{row_delimeter}
```

The first parameter to `bcp` is the name of the table or view to be copied. Choose `in` to copy the data from a file into that table and `out` to copy the data from that table to a file.

`bcp` can run in two modes: native and character mode. *Native* mode (set with the –`n` option) reads and writes output files in system native mode (that is, integers are stored as native integers and floats as native floats). Native mode files are not human-readable but can be significantly smaller and faster than those in character (ASCII) mode. In *character* mode, the files are human-readable. If you are copying in character mode, you need to specify a field delimiter with the –`t` flag. A rarely used character such as ~ or & makes a good delimiter. If you are using XML, however, you should use other delimiters. Sybase supports multicharacter delimiters—for example, `\t~\t` as a field delimiter and `\r\n` as a row delimiter. You could use a comma to create a *.csv* file this way, but if any of your data contains the delimiter character, your output will be corrupted. Files copied in character mode can be used interchangeably between operating systems while native *bcp* files scan only be used safely on systems with similar operating systems.

dsedit

The `dsedit` utility is a graphical editor for the Sybase *interfaces* file. The *interfaces* file maps server names to a hostname, port/socket number, and network protocol. The network protocol can also contain some other information for situations like fail over, and can contain mappings for the server to listen on more than one network interface or socket. Although the file can usually be edited in a text editor, administrators use `dsedit` because the *interfaces* files

on some operating systems use a packed notation that is not human-readable (you can read it but it makes no sense).

On Windows, the *interfaces* file is found in *$SYBASE/ini/sql.ini*.

The `tli` string in the Solaris *interfaces* file is a hex string containing port and IP addresses. If you have an entry that looks like the following:

```
SYBSRVR
 master tli tcp /dev/tcp \x000204018196c4510000000000000000
```

it can be interpreted as follows:

```
x0002 header info
0401  port number (1025 decimal)
81    first part of IP address  (129 decimal)
96    second part of IP address (150 decimal)
c4    third part of IP address (196 decimal)
51    fourth part of IP address (81 decimal)
```

This `tli` address is equivalent to the following entry in other operating systems, where the IP address of system *sybhost* is *129.150.196.81*:

```
SYBSRVR
    master tcp ether sybhost 1025
```

Required Environment Variables

The following environment variables are commonly set for Sybase:

`SYBASE`

Set this variable to the parent directory where you installed Sybase. This should not include the ASE version; that is the purpose of the `SYBASE_ASE` variable.

`DSQUERY`

Set this variable to the default server you wish to connect to.

`SYBASE_ASE`

Set this variable to the name of the subdirectory where the version of Sybase you're working with is located. For example, set it to `ASE-15_0` if you are working with 15.0, or to `ASE-12_5` for 12.5. The server software is located in `$SYBASE/$SYBASE_ASE`.

`PATH`

Set the path to include appropriate executable files.

`LD_LIBRARY_PATH`

Set this variable to help programs you compile find the appropriate libraries.

`SYBASE_OCS`

Set this variable to `OCS-15_0` if your client software is 15.0, or `OCS-12_5` if it is version 12.5. The client software is located in `$SYBASE/$SYBASE_OCS`.

You do not need to set these variables by hand when you log in because Sybase provides prebuilt environment files that set them. These variables are set in the *SYBASE.sh* and *SYBASE.csh* files that are located in your `$SYBASE` directory. You need to source one of the environment files before performing common operations like starting your database server.

Sybase client applications require `$SYBASE` to be set in order to find the *interfaces* file and may require `$LD_LIBRARY_PATH` for applications to find dynamic shared libraries.

The Power User's View

Power users of your database know enough to call you and ask questions about the terms in this section. They probably won't know anything more than these terms.

Server

A Sybase *server* generally refers to the Sybase database server, which is known as Sybase Adaptive Server Enterprise, or Sybase ASE. Each Sybase installation also includes a Sybase Backup Server and can include other Sybase server products, such as Sybase Monitor Server and Sybase Replication Server. All these servers use the same communication protocols and are, in fact, built using the same libraries (the API is published so that you can design your own servers if you so wish). The Sybase ASE Database Server uses symmetric multiprocessing. In other words, it can create multiple system processes that communicate through shared memory, and none of the processes has priority. When you start the database server, one or more `dataserver` processes will start. The number of `dataserver` processes that start is determined by the `max online engines` server configuration option. A normally functioning system also has a single `backupserver` process, which is responsible for the I/O of backups and restores. The `backupserver` process is a system process dedicated to performing fast backups and restores.

You can have as many servers on your machine as your system's resources can support. Each `dataserver` process requires a preallocated block of system memory, but the other servers are lightweight. For availability and performance reasons, it might make sense to have multiple

Sybase ASE servers running on the same machine, but it is common that each machine supports a single Sybase ASE server. The logic behind this is simple. With two servers, you can isolate a particular application and protect critical users from badly behaved applications and queries. On the other hand, a single 4-CPU ASE running on 16 GB of RAM has significantly greater throughput than two 2-CPU ASEs running with 8 GB RAM each.

Engine

When you start a Sybase ASE server, it executes one or more instances of the `dataserver` executable. Each running instance of `dataserver` is known as an *engine*. If you have a multiprocessor system with N processors, it is common to start either N or N–1 engines. The number of engines is set by the server configuration variable `max online engines`.

Database

A *schema* is a logically grouped collection of tables, indexes, stored procedures, and other database objects. A *database* is a collection of one or more schemas. A new server starts with several system databases. The system administrator creates one or more application databases on your server to contain the data and schemas of your business applications.

Because Sybase backups and restores run on a single database at a time, you should design your system so that all related objects are in the same database. In other words, a database usually represents an application, although your application can have several databases if it

can be split into several logically separated groups of objects.

Each account, after logging into your server, is placed in its default database (which is set on an account-by-account basis). Each account can be granted or denied access to objects in any of your databases. Sybase objects are normally accessed via the notation *database_name.owner.object_name*. Usually, the owner of each object is the database owner, and *owner* can be omitted. The shorthand to refer to objects in your current database is *owner.object_name* or simply *object_name*.

All servers contain the following system databases:

master

: The *master* database stores system configuration information such as logins and disk file mappings.

model

: This is a template database, and new user databases are copied from model.

tempdb

: This is a temporary space for sorts, temporary tables, and joins.

sybsystemprocs

: This database stores system stored procedures.

sybsystemdb

: This is used in ASE 15 to store transactions in progress and is used during recovery.

Servers also contain one or more user databases that contain your application information and can have pretty much any name.

Tip ✎

By convention, database names are in lowercase, often with an underscore in the name. It is also common to end databases with the suffix *db*. For example, *riskdb*, *logininfo_db*, and *tracedb* are all good database names.

Transaction

Sybase supports standard SQL transactions, guaranteeing that a set of one or more SQL statements succeed or fail together. Programmers create their own transactions by using the standard SQL statements `begin transaction` and `commit transaction`. Additionally, each `insert`, `update`, and `delete` statement is, by nature, its own transaction. A common example of a transaction is moving $100 from my account to your account. First, $100 must be subtracted from my account and added to your account. Both statements must either succeed or fail together. If the machine crashes between the statements, neither of the two statements should run, or else money will be lost.

Table

A *table* is a collection of related rows that all have the same attributes. Sybase tables can be partitioned, or spread across multiple page chains. When you partition a table, you allow that table to perform inserts faster.

Tip ✎

By convention, table names are in lower- or mixed-case, often with an underscore in the name. They also are singular. In other words, a table named *order* is preferable to a table named *orders*.

System Table

The *system table* stores system information such as logins, disk layouts, and system activity. In the Sybase ASE Architecture, all configuration information is stored in these system tables, which are well documented and can be accessed just like any other table. Sybase provides system stored procedures to read and modify this data.

System configuration information is stored in the master database. Database configuration information is stored in each database. All system tables start with the prefix *sys*. You cannot, by default, use Data Manipulation Language (DML) such as `insert`/`update`/`delete` statements on system tables. System tables can be manipulated only if you set the system configuration option that allows updates to system tables. This is set using the `sp_configure` system stored procedure. Needless to say, unless you understand exactly what you are doing, you should not edit your system tables by hand.

Tip ✎

Sybase system tables start with the prefix *sys*. Some system table names include *syslogins*, *sysdatabases*, and *sysdevices*.

Index

A database *index* is analogous to an index in a book: it allows Sybase to find data quickly. Sybase supports two types of indexes: clustered and nonclustered. A clustered index represents the sort order of the actual data pages (except with data-only locked, or DOL, page tables), and a nonclustered index represents a way to do index lookups to fetch the data.

Stored Procedures

A *stored procedure* is a precompiled set of SQL statements. Stored procedures provide a much faster way to perform common administrative tasks, queries, and other tasks.

The DBA's View

The Sybase-specific terms that follow are those that are of interest to a database administrator.

Page

A *page* is the smallest piece of data that can be moved within the database. Sybase's page size is set when the server is created, and cannot be changed. Most Sybase servers use 2 K pages, but larger page sizes can be used based on application requirements. A page can contain one or more rows.

Extent

An *extent* is a collection of eight pages that Sybase treats as one I/O unit. Disk reads are performed using extents. Whenever a table or index requires space, Sybase allocates another extent to the object.

Datafiles and Devices

Sybase stores its data in *datafiles*, which can be either raw partitions or cooked (filesystem) files. When datafiles are mapped into a Sybase system, they are known as *devices*. Once created, the system treats raw partitions and cooked datafiles identically. Prior to version 12.5, Sybase could not guarantee writes when using cooked files because of filesystem buffering. Version 12.5 introduced the concept of `dsync` devices, which are regular files to which writes are guaranteed. The `dsync` option is set when the `disk init` command is used to create a cooked file device. Another

type of datafile, which is just a normal file existing on a Unix *tmpfs* filesystem, permits data to exist in RAM.

I/O to filesystem devices can be faster than I/O to raw devices. This is not, however, always the case, and is dependent on the hardware vendor's implementation of the filesystem. There is, in fact, extra overhead in the filesystem, but that I/O is usually performed asynchronously. In general, you can assume that I/O to file devices is faster than I/O to raw devices, but because of the potential of hardware failure (and corruption resultant from that failure) in the past, Sybase recommends use of filesystems only when the data is totally recoverable. The advent of the `dsync` file option means that the risk of failure is no longer a concern, but the synchronous nature of file writes means that their performance is, in general, slower than that to raw partitions. If you are concerned about system I/O throughput, you should test I/O to raw partitions versus I/O to `dsync` cooked files and use whichever option is faster on your hardware.

Sybase directly controls all input and output to a raw devices or `dsync` cooked file rather than relying on the operating system to manage it. Most operating systems improve input/output performance by caching disk operations in memory and periodically writing them out to the physical disk. The problem with this is that Sybase assumes the I/O has already been written to disk when the I/O completes. If the operating system stops with items to write to disk, the disks will not match what Sybase thinks is out there, which can lead to corruption of the databases.

As mentioned earlier, cooked file I/O is faster than is raw file I/O, but should only be used if the data can be completely recreated. The database *tempdb* is used for temporary storage—for temporary tables and sorting of result sets. *tempdb* contains no permanent storage and, in

fact, is recreated whenever the server starts. You can gain performance by extending *tempdb* on cooked filesystem devices. The default configuration includes only 2 MB of *tempdb*, and it must be expanded for your system to function. *tempdb*, because it is recreated every time the server starts, cannot be lost in the event of a crash and can be placed on a regular filesystem file for performance.

> **Tip ✎**
>
> If your system supports *tempfs* files (filesystems in RAM), one possible mechanism to increase performance is to expand your *tempdb* onto this *tempfs*. This allows temp table writes to be done in RAM and not to disk, and can greatly improve performance.

Segment

A *segment* is a named collection of database devices available to a particular database. This is equivalent to Oracle's tablespaces and can be used to place database objects on specific devices. When a database is created, three segments (system, default, and logsegment) are created automatically. The *default* segment contains your data and indexes, the *logsegment* contains your transaction log, and the *system* segment contains your system tables. In addition to these segments, additional segments can be added using the `sp_addsegment` stored procedure. A database may contain several segments, which in turn can contain many objects, such as tables, indexes, and stored procedures.

A simple way to use segments to split up data I/O is to place the *default* segment (that contains data) on disk 1, the *logsegment* on disk 2, and to create a third index segment on disk 3. A typical `insert`, `update`, or `delete` statement

writes to all three of these disks; by using three disks, you split up I/O and increase performance.

> **Tip ✎**
>
> It is strongly recommended that you place your production databases on striped RAID disk arrays. Striping greatly increases system throughput, and RAID is necessary to protect your system from disk failures. Segments are normally used only if your system does not use disk striping.

Configuration File

The master database is a small database that contains system configuration information like disk layouts, server login information, and database configuration options such as the amount of memory to allocate to the database at startup. System-level configuration options are set using system stored procedure `sp_configure`.

Configuration options, when set, are automatically saved into a configuration file that is stored in *$SYBASE/$SYBASE_ASE/<SERVER>.cfg*. Whenever the configuration file is written, the old versions are archived. This allows administrators to review server parameters and gives the system an additional level of recovery (that is, you have the old values and can change them by editing the configuration file at the operating system level).

Transaction Log

A *transaction log* is a special system table (*syslogs*) that exists on the *logsegment* in every database. This table records all

changes to the pages in the database. Transaction logging cannot be turned on and off for Sybase databases. The transaction log is used by Sybase to guarantee transaction consistency, database consistency, and system recoverability.

By default, a transaction log stores all the information necessary to recover the database since the server started. It stores both the before and after images of the pages changed by your transactions. Since this obviously could be a lot of data, most of which is unnecessary because the changes have been written to disk—administrators must pick one of two strategies to clear the parts of the transaction log that have been physically flushed to disk. The first strategy involves auto-truncating the transaction logs table, and the second involves archiving the transaction logs to disk files.

Development systems and systems where you do not need intra-day system disaster recovery can use the automatic truncate option. This is identified by the database option `truncate log on checkpoint`, which deletes unnecessary records from the transaction log every few minutes. To set the `truncate log on checkpoint` option for database *my_db*, run the following commands:

```
$ isql –Usa –Ppassword –SMYSERVER <<EOF
  sp_dboption 'mydb,' 'truncate log on checkpoint,' true
  go
  use mydb
  go
  checkpoint
  go
  exit  EOF
```

This strategy is effective for development systems or other systems where your backups from the previous evening are sufficient for recovery. However, because you are truncating your transaction log, changes made on multiple days are unavailable.

Truncating your transaction log is therefore not acceptable for most production systems. On production systems, where you care about having multiple-day recoverability, transaction logs are normally saved to disk files prior to truncating them. These transaction log dump files can be used to guarantee system recoverability in the event of a catastrophic failure. Normally, if the system crashes, the database recovers all your committed transactions. The transaction log (even if it is being truncated) guarantees that all writes are flushed correctly to disk. When the server restarts, it first reviews the transaction log on disk for all transactions that have not been committed and written fully to disk. It then applies committed transactions, in order, to the server and rolls back uncommitted transactions. Once the database has finished its recovery process, it is brought online with fully consistent data.

For production systems, you must consider the case where the database server crashes and does *not* recover. In this rare case, you need to recover from backups. Recovering from the prior night's backup is normally acceptable for development systems because it is acceptable on these systems to lose a full day's worth of work when you go to backups taken the night before. On production systems, however, this is generally unacceptable. The Sybase recovery strategy for catastrophic server failures is to recover from full backup and apply the transaction log dump files—in order—to the database. This recovers the system to within a few minutes of the failure. If, for example, you were backing up transaction logs every 15 minutes, the maximum exposure of your data to a failure would be 15 minutes. It is typical for production servers to dump their transaction logs once every 5 to 15 minutes (depending on the business risks involved). Transaction log backups include only changes to the database and

therefore tend to be very quick, taking usually no more than a second or two to complete.

Of course, you must be careful where you back up your transaction log. It is considered bad practice to back up your transaction logs to the same physical disk that the actual database transaction log resides on. The reason for this is obvious. Disks fail often, so you would not wish to risk both your data and your backup by placing them on the same disk. For similar reasons, it is important that when you create your databases, you place the data and the transaction log on separate devices. The main reason for this is performance, but the disaster situation is greatly simplified if data and log are placed on separate devices.

What Happens When Transaction Logs Fill Up?

If a large transaction is run, or many transactions are running at the same time, the transaction log space for that database could become full. When this happens, all processing on that database is halted. Active transactions are suspended or aborted based on database configuration options. Additionally, a warning message is sent to active users that says that their process is out of space in either the *default* segment or *logsegment*. The wording of the error message should be inspected to discover which segment is full. If you are out of space in the *default* segment, you have filled up your database. You have no room left and need to either delete or add space. If you are out of space in the *logsegment*, you have filled up your transaction log.

The behavior of a database during these conditions is affected by the options set on that database, specifically the `truncate log on checkpoint` and `abort transaction on log full` options. The `truncate log on checkpoint` option

clears the transaction log every few minutes, but it clears only up to the first open transaction. It is possible that a single large transaction, or multiple long-running smaller transactions, could fill the transaction log. If this happens, and the `abort transaction on log full` flag is set to `true`, it rolls back the transaction that filled the transaction log. Your users will complain, but the system will be recovered. In this case, if the operations seemed well behaved and normal (and you can't tune them), you need to extend the transaction log. If the `abort transaction on log full` flag is `false`, the running transactions simply hang in `LOG SUSPEND` state until you clear some space for them to continue. You can see the status using the stored procedure `sp_who`, which shows these processes as blocked. This is a serious condition: your users' sessions have frozen, and they were *not* notified; their applications simply stopped.

Needless to say, `transaction log full` messages are a serious issue and should be dealt with promptly. There are a few things an administrator can do at this point:

- If you are out of space in the *default* segment, you can delete data from your tables to make room. This is the worst "solution" to this problem.
- You can add additional data or transaction log space using the `alter database` command. Be aware that once space is added to a database, it takes a lot of work to take that space back.
- If you are out of space in the *logsegment*, you should `dump` or `truncate` the transaction log manually. Your procedure here depends on your backup solution. Truncating the transaction log with `dump transaction with truncate_only` gets your users working again but is not something to do frequently because you will not be able to use the transactions that were stored in the

log for recovery purposes; you essentially threw them away.

If the transaction log is always filling up, the log is probably undersized, and additional space should be added. Fundamentally, this issue is serious and can usually be solved by allocating some disk space. The `dump transaction with truncate_only` command deletes the unused online transaction log and invalidates your used transaction logfiles as a disaster recovery solution (because some transactions that were applied to the database will be missing from the transaction log backup files). You should perform a full backup of the database as soon as possible.

Transaction log sizing

A normal rule is to size your transaction log 20 to 25 percent of the size of the database for a normally active database. This is not, however, a hard and fast rule. Fundamentally, you want to size the transaction log at twice the maximum space of insert, update, and delete statements that might occur in the transaction log dump interval. This actually implies that, as a percentage of data space, large databases with lots of historical data have a much smaller percentage of space used for transaction logs than databases that do batch updates. One hard and fast rule is this: if you ever run out of space in normal operations, you should increase the size of the log significantly. Disk space is cheap, but downtime and your time and effort are very expensive.

Here are examples of transaction log sizing. One database contains 1,000 GB of historical data updated by batch processes over the course of the day; its transaction log is only 5 GB. Another database contains 1 GB of data that is updated by a single large batch that completely refreshes

the data (deleting it all, inserting a new copy, and then performing multiple updates); its transaction log is 1.2 GB. Finally, another database contains a “normal” application with 200 MB of data; its transaction log is 50 MB. Note that in these three cases, transaction log size as a percentage of the data size varies dramatically.

The interfaces File

The *interfaces* file contains the information needed by Sybase client processes to connect to Sybase server processes. Specifically, it contains the hostname (or IP address), socket/port number, and a connection protocol (usually TCP). You may use the `dsedit` program to add entries to this file or modify the files by hand. The major difficulty in hand modification is that the default on Solaris systems is TLI TCP, where the address/port string is a binary string. The format of the file is shown in the following two examples taken from a Unix system:

```
$ cat interfaces
SYBPROD
   master tcp ether sybprod 5555
   query tcp ether sybprod 5555

PINKY
   query tli tcp /dev/tcp \x0002d6d8c0689915000000000000000
   master tli tcp /dev/tcp \x0002d6d8c0689915000000000000000
```

The SYBASE.sh and SYBASE.csh Files

The home directory of a Unix Sybase installation contains environment files (*SYBASE.sh* and *SYBASE.csh*) that you can use to set variables necessary for Sybase to function. Whenever the administrator logs in to the Sybase account, the first command she should run is to source the environment file.

Backup Server

The *backup server* is a separate Sybase server process designed to perform quick file I/O for database backups and restores. It is a required component if you wish to back up your systems.

The backup server can stripe the backup across media devices and do multiple-tape or multiple-file backups. The backup server can do two types of backups—a full database dump and an incremental transaction log dump. When the backup server does a full backup, all the database pages that contain data are backed up along with the current transaction log. Using this backup, a full restore of the database can be accomplished and contains all the changes up to the end of the backup.

The backup server can compress your database backups as they are being written. This compression is fast and effective. The data pages in the dataserver are mostly empty—making this compression a very nice feature. In fact, using compression level 1 (the lowest level of compression; it ranges from 1 to 10) creates a smaller backup file that is written significantly faster than backing up without compression.

Dump Device

Early versions of the Sybase database used logical dump devices to map backups to physical locations. You would then point your backup commands to these logical devices. The concept of a dump device is no longer used with modern Sybase systems because you can point your backups directly to specific files.

Hot and Cold Backups

While it is possible to shut down your Sybase database and perform a cold backup of the raw files that it uses (using `dd` or file copy commands), this is not normal practice. Sybase backups are normally run on live database servers, with the database software ensuring that the system recovers correctly. This has been true of Sybase since the system was released about 20 years ago. You never have to shut down your server to take a backup. Even on high-throughput 24×7 systems, the running of a Sybase backup should not be noticeable by your users. It is normal to run your backups during off-hours.

Protecting Your Database

When working with any database, the old maxim “an ounce of prevention is worth a pound of cure” holds true. Before covering details of how to perform a Sybase backup, we cover maintenance tasks that are normally performed on your servers on either a nightly or weekend schedule. The primary maintenance operations are `dbcc` and `update statistics`. `dbcc` checks the health of your server. `update statistics` updates the table distribution pages that Sybase uses to create query plans. `update statistics` is necessary to make your applications run well. These two steps are Sybase’s “ounce of prevention.” In addition to these `dbcc` tasks, you need to choose a transaction log archive strategy. If you follow these tasks, you will help maintain the database, keeping it running smoothly and ready for proper backups.

dbcc: The Database Consistency Checker

Even though Sybase’s dataserver products are very robust and much effort has gone into making them fault-tolerant, there is always the chance that a problem will occur. For very large tables, some of these problems might not show until very specific queries are run. This is one of the reasons for the database consistency checker, `dbcc`. This set of SQL commands can review all the database page allocations, linkages, and data pointers, finding problems and, in many cases, fixing them before they become insurmountable.

I strongly recommend running `dbcc` as part of a nightly preventive maintenance cycle. `dbcc` also should be run (if possible) before backing up a database, which adds an additional level of reliability to the backups. If a database is corrupt when it is backed up, the backup will also contain the corruption, and it is possible that the backup will not be restorable. `dbcc` can, however, be a time-consuming process,

taking an order of magnitude longer than a backup. A 200 GB database that takes 15 minutes to back up might take 3 hours to `dbcc`. While your system is not “down” while `dbcc` is running, `dbcc` degrades overall system performance and it can hold locks on tables that might impact application performance. Therefore, it is quite possible that your nightly maintenance cycle may preclude running `dbcc` every night. One option might be to run `dbcc` as part of a weekend maintenance cycle.

Since `dbcc` checks can be resource-intensive, consider adopting a strategy to take advantage of an object-level `dbcc` if you have a large database system (say, 500 GB and larger) and have high availability requirements. On a given day, run a certain number of `dbcc checktable` and `dbcc tablealloc` commands for a portion of the database. On subsequent days, run different tables. Over a period of days, you can accomplish a complete integrity check. For example, if your database has 200 tables in addition to the system tables, run a `dbcc` on each system table on night one, run `dbcc` against each of the first 50 of the user tables on night two, the next 50 the next night and so on, until at the end of five nights you have checked every table in the database. On the sixth night, you can begin the cycle again.

Building `dbcc` checks into your regular backup/maintenance schedule can ensure that you have a consistent, accurate database available at all times. `dbcc` is not an optional step if you care about your data.

Tip ✎

You should `dbcc` your database on some kind of routine basis. Simply running the command, however, is not sufficient. Search the `dbcc` output for error messages (usually running the Unix `grep` command or the Windows `find` command to look for `corrupt` and `error` is sufficient).

Standard/nightly dbcc checks

There are different types of checks `dbcc` can run on a database, and they differ in terms of runtime length, locking levels used, and completeness. Some check only data pages while others check index consistency and sort order. Which check should be run depends on the database size, the database access requirements, and the thoroughness required. Table 17-1 lists several `dbcc` checks that are usually performed as part of normal nightly maintenance.

Table 17-1. Types of dbcc checks

Keyword	Purpose	Access restrictions
`dbcc checkalloc`	Checks page allocations for all tables and indexes. Can fix some problems if allowed. Reports errors and the amount of space used. Very slow to run—lots of I/O, but very little locking.	Database owner only
`dbcc checkstorage`	Checks page storage.	Database owner only
`dbcc checkcatalog`	Checks the consistency of system tables in a database —very quick check. Outputs report on segments defined for database.	Database owner only
`dbcc checkdb`	Runs same checks as `checktable` but for all tables in a database.	Database owner only

If possible, you should run all these checks on each of your databases. The `dbcc` command is a Sybase-specific SQL command and is normally run using the `isql` command-line utility.

Here is the syntax for the `dbcc sql` statements:

```
$ dbcc checkdb
$ dbcc checkalloc [( database_name)] [, fix]])
$ dbcc checkcatalog [( database_name )]
```

The `fix` option of `dbcc checkalloc` controls whether `dbcc` should repair the problems it encounters. The `fix` option requires that the database be placed in single user mode and is therefore used only if prior runs of `dbcc` have detected problems (or if you have seen allocation errors in your Sybase error log). In fact, you should never run this command with the `fix` option unless you are instructed to do so by technical support or after looking up a previously discovered error message in the Sybase manuals. Before running `dbcc checkalloc, fix`, you must set the database option single user mode to `true`. Once it is run, only one user can access the database.

`dbcc checkstorage` is a newer mechanism to check database consistency. This command requires some basic installation to create a database named `dbccdb` that is used to store consistency check results. `dbcc checkstorage` is similar to `dbcc checkalloc` and `dbcc checkdb`, but it runs this check without holding locks on the underlying tables.

Reorgs

Reorganizing table data within Sybase is an optional step, but a step that can significantly enhance application performance. Sybase can leave extra whitespace when, for example, a variable length column is resized with an update statement (the original row size must be preserved in case the statement is rolled back). Additionally, repeated row

inserts and deletes on the same page can result in a nonoptimal distribution of rows. For example, if a data page can hold 10 rows, and you have deleted 9 of them, you could end up with a single row on the page. The page is not deallocated until the tenth row is deleted. This kind of allocation mismatch is not normally a big deal because rows are also inserted and pages combined, but if the size of your table changes too much, it can slow down the performance of selects. Because the data is stored in a b-tree, there is a point where the increased size of the data page chain causes an extra lookup by your queries. You can find the application taking 20 minutes to run one day and 35 minutes to run the next, simply because you added or modified some rows in a table. The Sybase `reorg` command compacts pages to their optimal size. This is the equivalent of dropping and reading your indexes but is done much more efficiently. The `reorg` command optimizes your page layout by copying rows around in a bunch of small transactions—a procedure that is usually quite fast and that does not hold table-level locks on your data so it can be done on running systems. Sybase provides four `reorg` commands:

```
reorg reclaim_space tablename [indexname]
[with {resume, time = no_of_minutes}]
reorg forwarded_rows tablename
[with {resume,time = no_of_minutes}]
reorg compact tablename
[with {resume, time = no_of_minutes}]
reorg rebuild tablename [indexname]
```

The `reorg compact` command is a combined run of `reorg reclaim_space` and `reorg forwarded rows`. With these `reorg` options, you can specify a maximum runtime so that they can be run during a specified maintenance window. The `reorg rebuild` command iterates through your data and optimizes everything. It cannot, however, be run within a specified time window. It is recommended that you choose either `reorg rebuild` or `reorg compact` and run it on all your tables at least once a month. Sybase requires only `reorg`s on DOL tables (DOL is a locking scheme).

Update Statistics

The `update statistics` *`tablename`* SQL command updates distribution statistics on your tables. The distribution information optimizes queries by permitting Sybase's cost-based optimizer to correctly select indexes to use in a query. Distribution information is kept on the table data (the clustered index) and on any nonclustered indexes for which you run the command. Once you have updated your statistics, future queries use the statistical information found about your table to choose the correct index. Existing stored procedures do *not* use the updated information. To allow stored procedures to access this information, you need to run the `sp_recompile` *`tablename`* procedure on each of your tables. `sp_recompile` tells the system that the next time a stored procedure that uses your table is run, it should be recompiled using the latest table statistics. It is normal to run `sp_recompile` on all your tables directly after you run `update statistics` on them.

> **Tip ✎**
>
> Commands like `update statistics`, `reorg rebuild`, and `sp_recompile` should be run on each table in your database and cannot be run at the whole database level. To find a listing of tables in your database, run the query `select name from sysobjects where type='U'`.

Configuration Audits

One important step to avoiding a disaster is to keep an up-to-date offline copy of your system configuration. A configuration audit can greatly decrease the time it takes you to restore in the event of a catastrophic system failure. You should have a copy of the commands that you used while

setting up your dataserver, a copy of the results of the system stored procedures `sp_helpdb` and `sp_helpdevice`, and a `bcp` output of the most important system tables in the master database, including:

sysusages
syslogins
sysremotelogins
sysloginroles
syssrvroles
sysdatabases
sysdevices
syscharsets
sysconfigures
sysservers
sysremotelogins
sysresourcelimits
systimeranges

In addition, maintain a hard copy by printing the output of the following queries:

```
$ cat backup_systemtables.sql
exec sp_helpdb
go
exec sp_helpdevice
go
select * from sysusages order by vstart
select * from sysusages order by dbid, lstart
select * from syslogins
select * from sysloginroles
select * from sysdatabases
select * from sysdevices
select * from syscharsets
select * from sysconfigures
select * from sysservers
select * from sysremotelogins
select * from sysresourcelimits
select * from systimeranges
go
```

These queries can be scheduled or run using a command such as:

```
$ XSQL backup_systemtables.sql
```

XSQL is explained later in this chapter. You should also ensure that you have an easily available backup of the Sybase software directory. This should include copies of your configuration file.

Implement Mirroring and Disk Striping

Mirroring, either at the server level or at the operating system level, can provide nonstop recovery in the event of media failure. I strongly recommend implementing mirroring at the operating system level. Sybase also supports software mirroring. If you do not have hardware mirroring, you should read "Mirroring Database Devices" in the *Sybase System Administration Guide*. When implementing a RAID solution for your databases, it is important to set up disk striping. Databases are often I/O-limited; and disk striping, by nature, optimizes I/O by spreading it across multiple disks.

How to Back Up Your Servers

Sybase backups are normally run while the database is up and active. The transaction log mechanism in Sybase guarantees system consistency. In Sybase, the SQL command to run a full database backup is `dump database` and the command to run a transaction log dump is `dump transaction`. The `dump` command does a byte-by-byte copy of all populated pages in the database. Both the `dump` and `restore` commands are performed at the database level.

There are two normal ways to back up your databases. The first is to back up your databases every night and to set the database option `truncate log on checkpoint`. In this case, you do *not* run transaction log dumps. This is the normal scheme for development databases—databases in which your data can be rebuilt or where the loss of multiple day data changes to your data is acceptable. The other option is to run

full database backups every night and incremental backups every 5 to 15 minutes. In this case, `truncate log on checkpoint` is not set. The incremental backup is performed by running the SQL command `dump transaction to "/dat/myfile"`. This creates a file (*/dat/myfile* in this example) that contains your transaction dumps and deletes all transaction records that have been committed and are marked as having been flushed to disk.

> **Tip ✎**
>
> You should name your dump files so that they sort alphabetically. For example, you can use the convention *yyyymmdd.hhmmss*. If you name them using such a format, it will be easier to work with them if they need to be restored.

The `dump` and `load` SQL statements are not actually run by the dataserver. These commands are run by a separate process known as the backup server. You must therefore ensure that the backup server is running at all times.

Syntax of the dump statement

In its simplest form, a full database backup looks like this:

```
> dump database somedb to "file"
```

In its simplest form, a transaction log backup like this:

```
> dump transaction somedb to "file"
```

The file in these commands can be a tape device, a disk file, or a dump device defined in Sybase using the stored procedure `sp_adddumpdevice`. (Dumping to tape devices and logical dump devices is nowadays often rejected in favor of backing up directly to disk files.) Example 17-1 shows sample dump commands.

Example 17-1. Sample dump commands

```
1> dump database mydb to '/sybase/backups/mydb.990312.bck'
2> go
Backup Server session id is:  20.  Use this value when executing
the
'sp_volchanged' system stored procedure after fulfilling any volume
change
request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file
/sybase/backups/mydb.990312
.bck.
Backup Server: 6.28.1.1: Dumpfile name 'mydb9909106EF4   ' section
number 0001
mounted on disk file '/sybase/backups/mydb.990312.bck'
Backup Server: 4.58.1.1: Database mydb: 338 kilobytes DUMPed.
Backup Server: 4.58.1.1: Database mydb: 344 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database mydb: 352 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database mydb).
1> dump transaction mydb to '/sybase/backups/mydb.tlogdmp'
2> go
Backup Server session id is:  23.  Use this value when executing
the
'sp_volchanged' system stored procedure after fulfilling any volume
change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'mydb9909106F49   ' section
number 0001
mounted on disk file '/sybase/backups/mydb.tlogdmp'
Backup Server: 4.58.1.1: Database mydb: 16 kilobytes DUMPed.
Backup Server: 4.58.1.1: Database mydb: 20 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database mydb: 24 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database mydb).
```

Backup striping and compression

The Sybase dump command can split the backup files between several files, a process called *striping*. The dump command also supports integrated file compression. This native compression should always be used because compressed backups are faster and fit into smaller files.

Here’s the syntax for striping:

```
> dump database somedb to "file2"
> stripe on "file2"
> go
```

Here's the syntax to use compressed file backups:

```
dump transaction somedb to "compress::number::file2"
go
```

The number refers to the compression level and ranges from 0 to 9 (where 0 indicates no compression). I recommend using compression level 1 for your backups; this represents the fastest level of compression. The performance and size gain made by compressing your dumps at level 1 is significant when compared to uncompressed backups. There is often a lot of whitespace in your raw database files—whitespace ideally suited to backup compression.

If you are backing up to a file using both striping and compression (recommended), the syntax of your command will look like this:

```
dump database mydb
to "compress::1::/dumps/mydump-stripe1"
stripe on " compress::1::/dumps/mydump-stripe2"
stripe on " compress::1::/dumps/mydump-stripe3"
stripe on " compress::1::/dumps/mydump-stripe4"
go
```

In all cases, if you are unsure about the specific use of a command, please refer to the Sybase system administration manuals for additional guidance.

Have a Run Book

Because most data centers have more than one dataserver and database, having clear documentation helps prevent mistakes. I strongly recommend that you keep a *run book* that contains configuration and batch job scheduling information on all your servers. A run book need not be a paper copy of your configuration; an electronic run book would be a better idea. Simply dump the output of common system procedures

(`sp_helpdevice`, `sp_helpdb`) into text files and keep a few weeks' worth. Having information about your historical system layout can save time in the event of a problem (it gives you more data for analyzing how the problem started) and is critical in the event of a disaster.

Backup Automation Through Scripting

This section gives you a simple mechanism to back up your Sybase servers. These examples are written in the Bourne shell and give you a minimal set of required server management functionality. They are provided for illustrative purposes, but the scripts can easily be used for production backups of your systems.

Backup Automation Basics

It's a good idea to create a scripts directory on each of your servers. For example, you could call this directory */export/home/sybase-scripts*. The first script in this directory is called *XSQL* and looks like this:

```
$ cd /export/home/sybase-scripts
$ cat XSQL
#!/bin/ksh
. /export/home/sybase/SYBASE.sh
/export/home/sybase/OCS-12_5/bin/isql -Usa –SMYSERVER
     -Psecret -i/export/home/sybase-scripts/$1
     -o/export/home/sybase-scripts/$1.out
echo "Script Completed" >> $1.out
date >> $1.out
```

This function of this script should be obvious. The *SYBASE.sh* file is sourced to set up environment variables, a necessary step if you are running a job out of `cron`. The *XSQL* script takes one argument, which is a script file, and runs it, placing the output in a file that is the name of the script file with *.out* appended.

Let's say you have a database named *mydb* to back up. You could create a file called *backupmydb4way.sql* that looks like the following:

```
$ cat backupmydb4way.sql
set nocount on
```

```
select getdate()
go
DUMP DATABASE mydb
to "compress::1::/dumps/mydump-stripe1"
STRIPE ON " compress::1::/dumps/mydump-stripe2"
STRIPE ON " compress::1::/dumps/mydump-stripe3"
STRIPE ON " compress::1::/dumps/mydump-stripe4"
go
select getdate()
go
```

This creates four files in */dumps* that represent your backups. Schedule `/export/home/sybase-scripts/XSQL /export/home/sybase-scripts/backupmydb4way.sql` using `cron` to create backups at night.

A simple update stats script

`update statistics` is a bit more complicated to run because it must be run on each table in your database. The following script is an example of how to run a SQL command on every table in a database. It runs `update index statistics` *`tablename`* followed by `exec sp_recompile` *`tablename`* on all tables in your database.

```
$ cat updstatsmydb.sh
#!/bin/sh
. /export/home/sybase/SYBASE.sh
ISQL="$SYBASE/$SYBASE_OCS/bin/isql -Usa –Psecret -SMYSERVER -
w160"
SQLFILE=/tmp/updstats.sql
LOGFILE=\Qdate '+updstats2_%Y%m%d'\Q

cat << EOF | $ISQL  | sed -e 's/-//g' -e 's/([0-9]* [rows
affected]*)//g' > $ SQLFILE
select "select convert(varchar,getdate())+ ' upd stats
"+name+"'"+char(10)+"go"+char(10)+"update index statistics
mydb.dbo."+name+char(10)+"go"+"exec sp_recompile
+name+char(10)+"go" from mydb.dbo.sysobjects where type = 'U'
go
EOF
$ISQL -i$SQLFILE -o$LOGFILE
```

This Bourne shell script file runs the commands `update index statistics` and `sp_recompile` on all the tables in the database *mydb*. The */tmp/updstats.sql* script is created and then executed at the last line of the script. The */tmp/updstats.sql* file contains `print` statements that print the time and table name followed by an `update stats` for each table in your database.

Note that `char(10)` happens to be a newline. The `sysobjects` table in each database contains a list of all objects in your database. Objects of type `U` are tables. The */tmp/updstats.sql* file created by the script looks like this:

```
select convert(varchar,getdate())+ ' upd stats tbl1'
go
update index statistics mydb.dbo.tbl1
go
exec sp_recompile mydb.dbo.tbl1
go
select convert(varchar,getdate())+ ' upd stats tbl2'
go
update index statistics mydb.dbo.tbl2
go
exec sp_recompile mydb.dbo.tbl2
go
<etc>
```

A sample transaction logfiles backup script

You should dump compressed versions of your transaction logfiles, but you need not back these files up using striping. It is also recommended that the script you use to run `dump transaction` back the files up using 24-hour time stamping. Essentially, this means that the file should have date and time in the format `yyyymmdd.hhmmss` (year, month, day, hour, minute, second). Using this format ensures that a standard file listing shows your transaction logfiles in order. Here's an example of a command that does this:

```
DTIME=\Qdate "+Y%m%d.%H%M%S"\Q
echo "dump transaction somedb to">> /tmp/dfile.sql
```

```
echo "\"/export/home/sybase-dumps/somedb.$DTIME.trn\"" >>
/tmp/dfile.sql
echo go >> /tmp/dfile.sql
XSQL /tmp/dfile.sql
```

Schedule backups

You will need to schedule backups of your database using `cron` or the Windows task scheduler. Here is an example of an entry using `crontab`:

```
0 2 * * * /export/home/sybase-scripts/XSQL backupmydb4way.sql
```

This runs your simple backup at 2:00 a.m. every day. A file */export/home/sybase-scripts/backupmydb4way.sql.out* contains a log of the last run. You will need to run a backup of all your databases with the exception of *model* and *tempdb*. You can get a listing of your databases by running the stored procedure `sp_helpdb`.

Mailing crontab results

Put a *.forward* file in the Sybase home directory so that errors are mailed to your email account. This file should contain one email address per line.

Logical Backups

The `sybase dump` command backs up a single database at a time. If a backup of a single table is needed, this command cannot be used. The `dump` command cannot be used to copy a database between servers running different operating systems because the backup files are in an operating system–dependent format. The logical (table-level) backup utility `bcp` backs up only parts of a database or copies data between systems on different operating systems.

bcp provides another key feature. Because it refers to the internal structure of the database, it is an excellent tool to confirm whether there is corruption while also backing up the data. The bcp command displays an error if it has any problems reading the data.

The two drawbacks to bcp are:

- bcp takes much longer than dump to back up a whole database.
- bcp can back up only tables and views. Other database objects (stored procedures, triggers, and so on) must be exported using a different method.

Performing a logical backup

bcp can create an export of the entire database, or parts of it. Here are some examples. To copy a database in, use a command like the following:

```
# bcp mydb.dbo.zyx in /sybackups/zyx.bcp -c -S SERVER -U sa -P mypassword
Starting copy...

192 rows copied.
Clock Time (ms.): total = 1000   Avg = 5    (192.00 rows per
sec.)
```

To copy a database out, use a command like this:

```
# bcp master.dbo.sysusages out /sybackups/sysusages.bcp -c -S SERVER -U \
   sa -P mypassword
Starting copy...

9 rows copied.
Clock Time (ms.): total = 1000   Avg = 111    (9.00 rows per
sec.)
```

Performing a logical restore

The `bcp` utility can run in fast or slow mode when importing data into a table. If the table does not have indexes or triggers, `bcp` uses a fast mode to insert the data. This mode is faster because changes are not logged into the transaction log. In fact, after this is done, a transaction log dump is invalid until a full database dump is done. Fast mode is used when a large amount of data must be inserted into a table. Remember to run a full database dump when the `bcp` completes.

If an index or trigger exists on the table, the slower, logged mode of `bcp` is used. Be careful when inserting large amounts of data. One prerequisite before using `bcp` to restore data into a table is to make sure the table exists. `bcp` does not create objects; it just transfers data in and out of the database. `bcp` is also handy when you need to import data from a delimited data source such as a comma-separated set of values (CSV) file. The latter topic is beyond the scope of this book, but more information can be found in the Sybase utilities manual. Here, we will discuss how to store table data in a file and how to restore it back into a database.

The main syntax difference between using `bcp` to import data into a table versus out of a table is changing the word `out` to `in`. All other parameters are exactly the same. A few important factors that do not come into play when using `bcp` to export data are `-e` *`errorfile`* and `-m` *`maxerrors`*. The `-m` option specifies the maximum nonfatal errors `bcp` allows before quitting. The `-e` option specifies the error file where the data rows from the `bcp` file that caused the nonfatal errors are stored for later review. (In both cases, any error messages display at the terminal.) These parameters allow the `bcp` command to continue after encountering nonfatal errors and to record the errors for later correction.

When the bcp file is created using the -c option, the records are stored in a character-based, transportable record format. This is very handy when adjustments need to be made to the data before using bcp to import them back into a database. This format allows the use of a simple text editor to make the changes. For example, if the third column in the bcp file represents department names, and one of the names has been changed, use an editor to open the file and do a replacement of the old name with the new name. Also, if a number of records are failing, and they are preventing the bcp -in from continuing, edit the file, and delete the problem lines. If you are using the –c option to bcp, you will probably also need to specify the –t (field terminator) argument. See the earlier section on bcp regarding the use of the –t argument.

One consideration when using bcp to restore system tables is that the database always contains certain default entries —such as the *syslogins* entry sa (an abbreviation for system administrator). When trying to restore using a bcp of this table, the bcp will fail because of the unique index on this table. To get around this, copy the *syslogins* bcp file and remove the entries from the copy that is already in the *syslogins* table. Once these records are gone, the bcp import will succeed. You can also set the batch size option to 1 row with –b1. This causes duplicate rows to be ignored.

Auditing using bcp

The system tables suggested for auditing should be copied out of your server on a weekly basis. Use the -c option enables you to view the data in the datafiles even if the dataserver is down.

```
$ cat sysaudit_bcp.ksh
cd /sysbackups
bcp master.dbo.sysusages out sysusages.bcp –c –T~ -SSYB_MYDB -Usa
```

```
–Pmypassword
bcp master.dbo.syslogins out syslogins.bcp –c –T~ -SSYB_MYDB -Usa
–Pmypassword
bcp master.dbo.sysloginroles out sysloginroles.bcp –c –T~ -
SSYB_MYDB -Usa -Pmypassword
bcp master.dbo.sysdevices out sysdevices.bcp –c –T~ -SSYB_MYDB -
Usa –Pmypassword
bcp master.dbo.sysdatabases out sysdatabases.bcp –c –T~ -
SSYB_MYDB -Usa –Pmypassword
bcp master.dbo.syscharsets out syscharsets.bcp –c –T~ -SSYB_MYDB
-Usa –Pmypassword
bcp master.dbo.sysconfigures out sysconfigures.bcp –c –T~ -
SSYB_MYDB -Usa –Pmypassword
bcp master.dbo.sysservers out sysservers.bcp –c –T~ -SSYB_MYDB -
Usa –Pmypassword
bcp master.dbo.sysremotelogins out sysremotelogins.bcp –c –T~ -
SSYB_MYDB -Usa –Pmypassword
bcp master.dbo.sysresourcelimits out sysresourcelimits.bcp –c –T~
-SSYB_MYDB -Usa –Pmypassword
bcp master.dbo.systimeranges out systimeranges.bcp –c –T~ -
SSYB_MYDB -Usa –Pmypassword
```

Physical Backups with a Storage Manager

Many commercial backup utilities have a completely integrated backup solution for Sybase databases. They have an interface that communicates directly with Sybase's backup server. This interface provides additional functionality by enabling you to create, schedule, and execute the backups via a graphical user interface (GUI). It lets you run the backups and restores with just a few mouse clicks.

Sometimes, these utilities are on the same system as the dataserver; at other times, only a client part of the application will be on the dataserver system, and a separate backup application will be on a networked system. No matter how it is set up, these applications provide an easier and reliable backup solution.

In addition to providing support for physical backups, the commercial utilities described in Chapter 8 can also provide support for logical backups. They add additional functionality to `bcp`, such as easy scheduling and a GUI frontend, and allow the logical backups to be stored on remote backup devices.

Because of the strong integration with Sybase's backup system, sometimes the only change seen on the Sybase side might be to the destination or source device used. This allows easy merging of the new product with existing backup procedures. For example, where the physical tape device had been */dev/nrmt0* in a script, the new device might have the form:

```
"3rdPartybackup::'99.3.10.04.00.master.full' "
```

The `3rdPartybackup` phrase tells the Sybase backup server that the backup should go to the third-party utility. The `99.3.10.04.00.master.full` phrase represents the time,

database, and what type of backup is being done. In this way, information about the backup can be sent to the backup utility so it knows how to record and store the backup. This value would be used both in the backup and recovery commands in Sybase.

Recovering Your Database

The syntax of recovering your backups is identical to that of backing them up. The keyword `backup` is replaced with the keyword `restore` and the keyword `to` is replaced with `from`.

Recovering from a Disaster

Disaster recovery is different from a normal restore. You do, however, need a plan in the unlikely case that a catastrophic failure occurs. You must plan for location outages, system outages, component outages, and database corruption. You should have an idea of how to recreate your system and recover each of your databases from backups.

The fundamentals of disaster recovery are not covered here (see Chapter 24 for treatment of this topic). You will need access to your configuration audits and database backups on whatever system you eventually want to restore from. Be sure to think about the basics of your disaster plan—considerations like ensuring that the tape backup of your database dump files occurs *after* your database dumps complete.

The most common systems failure you will face is a disk drive failure. The most obvious way to protect your systems is therefore to always use disk systems protected by RAID. Most large organizations require RAID on all systems; disk failures are too common a condition to ignore, and there are significant administrative and downtime costs to these failures even if no data is lost.

The case in which your database server fails catastrophically is rare but can happen. To deal with this situation, you should be able to restore your system from

scratch. This includes having all necessary information and software to do this recovery.

Restoring from Backups

Loading a Sybase database from a database dump file is not a quick process. You can estimate that it will take between 1 and 2 hours per 100 GB of database space, so you should plan accordingly. You will also need to budget time to apply transaction log backups if that is necessary. The database into which you restore your backups needs to be created in the exact same order and size as the original database. There is no easy stored procedure available in the default installation of Sybase that details this information for you, but all major Sybase tools (including Sybase Central—the enterprise manager that Sybase ships) can rebuild these statements for you. If you are trying to restore a backup into a second server or into a server that has no existing database of the appropriate name, you need to recreate the database using commands like the following:

```
C:> create database mydb on mydefseg = 40, mydefseg = 40
       log on mylogseg = 10
C:> alter database mydb on mydefseg= 40
```

Tip ✎

I also distribute a widely used free set of Sybase and SQL Server extended stored system procedures from my web site http://www.edbarlow.com. The stored procedure `sp__revdb` *dbname* can be used to reverse-engineer the layout of your databases; you can download it from my site.

Once the database creation is complete, the full dump of the database needs to be applied using the `load` command, as shown in Example 17-2. The `load` command is identical in structure to the `dump` command.

Example 17-2. Sample load database command

```
1> load database mydb from '/sybase/backups/mydb.990312.bck'
2> go
Backup Server session id is:  26.  Use this value when executing
the
'sp_volchanged' system stored procedure after fulfilling any
volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'mydb9909106EF4   '
section number
0001
mounted on disk file '/sybase/backups/mydb.990312.bck'
Backup Server: 4.58.1.1: Database mydb: 17926 kilobytes LOADed.
Backup Server: 4.58.1.1: Database mydb: 19462 kilobytes LOADed.
Backup Server: 4.58.1.1: Database mydb: 19470 kilobytes LOADed.
Backup Server: 3.42.1.1: LOAD is complete (database mydb).
Use the ONLINE DATABASE command to bring this database online;
SQL Server
will not bring it online automatically.
```

As noted in the output, you will need to run the `online database` *dbname* TSQL command to activate your database after a load. If there are no transaction logs to apply for this database, you can immediately run `online database` *dbname* to activate the database. If you wish to apply transaction logs, you should apply them first and then run the `online database` command.

To apply a transaction log to the database, use the `load transaction` command. To restore the transaction logs for the database *mydb* in the preceding `dump` example, enter the command shown in Example 17-3.

Example 17-3. Sample load transaction command

```
1> load transaction mydb from '/sybase/backups/mydb.tlogdmp'
2> go
Backup Server session id is:  28.  Use this value when executing
```

```
  the
  'sp_volchanged' system stored procedure after fulfilling any
  volume change
  request from the Backup Server.
  Backup Server: 6.28.1.1: Dumpfile name 'mydb9909106F49   '
  section number
  0001 mounted on disk file '/sybase/backups/mydb.tlogdmp'
  Backup Server: 4.58.1.1: Database mydb: 24 kilobytes LOADed.
  Backup Server: 3.42.1.1: LOAD is complete (database mydb).
  Use the ONLINE DATABASE command to bring this database online;
  SQL Server
  will not bring it online automatically.
```

Repeat the transaction log loads until there are no more transaction logs to apply. It is strongly recommended that you dump your transaction logfiles to a filename that contains a timestamp in the format `yyyymmdd.hhmmss`. If you do this, you can list the files, create a command batch, and then restore the transaction logs in order.

The online database command

When this process is done, the database has been restored completely and should be brought online using the `online database` *mydb* command.

Restoring to a specific time

Sybase can also use saved transaction logs to restore a database to a specific point in time. So, if someone added important data at 12:00 p.m. and then someone else deleted this data by accident at 2:00 p.m., you can restore from a backup and apply saved transaction logfiles to restore the database to exactly 1:59 p.m. Data is back, and the users are happy.

> **Tip ✎**
>
> To recover deleted production data without having production downtime, you can load backups of the database into development, load the saved transaction logfiles, and then use a tool such as `bcp` to copy the data from development to production.

To specify the time to which the database should be restored, use the `until_time` parameter. This parameter takes a single value of a time and date in the default format for the dataserver. For example, to restore a database up to April 1, 2007 at 12:34:32:650 a.m., first apply the full database dump, then apply all transaction log dumps up to the one that contains the stop time. With this dump, add `until_time`, like this:

```
load transaction mydb
from '/sybase/backups/mydb.tlogdmp'with until_time = 'April 1, 2007 at
12:34:32:650AM'
```

Restoring from compressed backups

As noted in the backup section, you can back up your data to compressed files using the syntax `compress::` *`compression_level`* `::filename`. One final difference between the `dump` and `load` commands is that the `load` command does not need the compression level specified. The `load database` command can figure out the compression level from the backup file. The syntax for loading compressed files is therefore `compress:::filename`.

Common Sybase Procedures

This section describes a full diagnostic and recovery procedure for your system. We'll begin with the basics and work toward more complicated recovery procedures. First we'll cover some commonly used procedures—procedures that are referenced by the rest of this chapter.

Procedure 1: How to Start Sybase

In the *$SYBASE/$SYBASE_ASE/install* directory you will find your server's run files. Run files are simple shell scripts that begin with the pattern *run_* and usually are named *run_<SERVERNAME>*. They contain shell commands that are used to start your Sybase servers with appropriate command-line arguments. Backup server run files are normally named *run_<SERVERNAME>_BACK*, and monitor server run files are named *run_<SERVERNAME>_MON*. A default installation includes several of these run files, one for your database server, one for your backup server, and one for other servers in the Sybase product suite.

Example 17-4 shows a sample run file for a Unix system.

Example 17-4. Sample run file

```
#!/bin/sh
#
# SQL Server Information:
#  Name:                            SYB_TITANIA
#  Master device:                   /sybdata/master.dbf
#  Master device size:              10752
#  Errorlog:
/opt/sybase/logs/SYB_MYDB.errorlog
#  Interfaces:                      /opt/sybase
#
/opt/sybase/bin/dataserver -d/sybdata/master.dbf -sSYB_MYDB \
-e/opt/sybase/logs/SYB_MYDB.errorlog -i/opt/sybase
```

On a Unix system, a server is started with the command `startserver -f` *`run_FILE`*. The `startserver` command is a very simple binary executable that basically runs the file in the background.

> **Tip ✎**
>
> In Unix, Sybase Server should always be started using the `sybase` account. You should not start using the `root` account.

Procedure 2: How to See Whether Your Server Is Alive

The obvious way to see if your server is OK is to connect to it with a utility like the `isql` program. `isql` works with both the Sybase server and the Sybase Backup server. Here is an example of this process, connecting to a dataserver *SYB_MYDB* and to a backup server *SYB_BACKUP*:

```
# isql -S SYB_MYDB -U sa -P passwd
1> quit
# isql -S SYB_BACKUP -U sa -P passwd
1>quit
```

If either of these processes is not up, or if you entered an invalid `sa` password, an error message like the following appears:

```
Operating-system error:
        Connection refused
DB-LIBRARY error:
        Unable to connect: SQL Server is unavailable or does not
exist.
```

If you suspect there is a problem with your server or you cannot log in, you can check that the system processes are running. Sybase processes can be checked either by

running the standard `ps` command or by running a Sybase-provided shell script file called *$SYBASE/$SYBASE_ASE/install/showserver*. *showserver*, in fact, calls the `ps` command. A running system should have one or more `dataserver` processes and one `backupserver` process.

Example 17-5 shows both commands and their outputs from a Linux system.

Example 17-5. Sample ps and showserver outputs

```
# ps ax       # or ps auxww | grep sybase
  191  ?  S    0:00 in.telnetd
   94  ?  S    0:00 /usr/sbin/portmap
  114  ?  S    0:00 /usr/sbin/atd
  157  ?  S    0:00 sh /opt/sybase/install/run_SYB_BACKUP
  160  ?  S    0:00 sh /opt/sybase/install/run_SYB_TITANIA
  164  ?  S    0:00 /opt/sybase/bin/dataserver -
d/sybdata/master.dbf
                      -sSYB_MYDB
  168  ?  S    0:00 /opt/sybase/bin/backupserver -SSYB_BACKUP -
e/opt/sybase/logs

# $SYBASE/$SYBASE_ASE/install/showserver
sybase     164  0.1 10.4 16224  6592  ?  S    22:18   0:00
  /opt/sybase/bin/dataserver  d/sybdata/master.dbf -sSYB_MYDB
  -e/opt/sybase/logs/SYB_MYDB.errorlog -i/opt/sybase
sybase     168  0.0  6.4  6660  4092  ?  S    22:18   0:00
 /opt/sybase/bin/backupserver
 -SSYB_BACKUP -e/opt/sybase/logs/SYB_BACKUP.errorlog -
I/opt/sybase/interfaces
 -M/opt/sybase/bin/sybmultbuf -Lus_english -Jiso_1  -
c/opt/sybase/backup_tape
```

If the `backupserver` or `dataserver` processes are not up and running, follow the appropriate operating system commands to start them.

On Windows, Sybase runs these processes as services—you can see their availability by looking in the Control Panel under Administrative Tools.

Procedure 3: How to Shut Down Your Server

Sybase offers several ways to shut down your server. The primary mechanism to shut down your server is through the T-SQL command `shutdown`. When this command is run, the database disables all logins except the login for the system administrator, performs a checkpoint on each database, and waits for all currently running statements or procedures to complete. If any user connections have not completed their transactions, the shutdown waits for them to complete. If you do not want to wait for these transactions to complete, you can try the command `shutdown with nowait`. The `nowait` option stops all transactions immediately and continues with the shutdown, but because there could be uncommitted transactions in progress, recovery time is increased. It's best to check with your users before doing this. (Otherwise, there might be one less DBA in the world when the users track down the person responsible for pulling the rug out from under them.) Sybase 12.5.4 introduced a wait time for the `shutdown` command, so that it will wait for a period of time, then do a `shutdown nowait` if the system hasn't shut down by then.

The first thing to try—normal shutdown

A normal shutdown is the ideal way to shut down your database server. It minimizes server restart times by allowing all active transactions to complete and then flushing all unwritten pages to disk by running a checkpoint in each database. Since most transactions are short, a normal shutdown should complete within a few seconds. All transactions are written to disk and marked as committed, so the server recovery process is very fast.

```
1>shutdown
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
```

The second thing to try—shutdown with nowait

`shutdown with nowait` kills running processes and then shuts down your dataserver. Transactions that have not completed abort and are rolled back when the dataserver starts up. This increases startup times; because checkpoints were not run and data not flushed to disk, the server must reconstruct active transactions from the transaction log while starting up. The server goes through all transactions you applied and applies only those that were committed. This method is used quite frequently to shut down the server when `shutdown` does not respond quickly.

```
1>shutdown with nowait
Server SHUTDOWN by request.
The SQL Server is terminating this process.
DB-LIBRARY error:
        Unexpected EOF from SQL Server.
```

The third thing to try—kill -15 on the dataserver

There are rare times when you cannot log in to your dataserver to shut it down. Sybase has coded the `dataserver` process to interpret signal 15 (sent by a `kill –15`) as a `shutdown with nowait`. In other words, the server runs a `shutdown with nowait` when it gets a `kill –15`. This can, therefore, be considered a safe way to stop your `dataserver`, although active transactions are rolled back when the server starts up again, and server restart time is increased.

What you should NEVER do—kill -9 on the dataserver

Commercial databases are very robust these days. In all likelihood, your server will survive a `kill –9` on the `dataserver` process and recover just fine. But it would be nice to guarantee server recovery, something you cannot do when you run `kill –9`. So run this command on your `dataserver` processes only as an *absolute last resort*. A `kill –9` will remove the process from the process table and does not allow that process to run its embedded cleanup routines. Pages are swapped out of memory; this can be considered the same as flipping the power switch on your system. Always try the other mechanisms to shut down your server before you try this.

On Windows

On Windows, you should first try the normal shutdown commands. If these do not work, you can stop a server by stopping the appropriate Windows service using the Services option in the Control Panel.

Procedure 4: How to Set Server Configuration Options

Server-level options are set using the stored procedure `sp_configure`. Running `sp_configure` without parameters lists current configuration options. To set options, users run `sp_configure configoption, configvalue`. Most Sybase configuration options are dynamic and affect your running server immediately (although a few require a reboot of the server).

Procedure 5: How to Set Database-Level Options

Each database in your system has several associated configuration options. Examples of these options include `truncate log on checkpoint` (which removes the unused part of the transaction log every few minutes), `abort transaction on log full` (which aborts running transactions if the transaction log fills; the default behavior is to suspend the process), `select into/bulkcopy` (which permits certain operations to run faster by not putting them in the transaction log), `single user mode`, and `dbo use only`. The following procedure sets a database option *optionname* in database *mydbname*:

```
sp_dboption mydbname, "optionname," true
go
use baddbname
go
checkpoint
go
```

Procedure 6: How to Run a Query

As mentioned earlier, the `isql` command is not very feature-rich. One of the features it does not support is easy output redirection to a file. There are two options for this. The first is the free program `sqsh` (created by Scott Gray and maintained by Michael Peppler), which is basically an advanced version of `isql` that supports command history and output redirection. The next is `ASEisql`, a free graphical utility. You can also, of course, use SQL Advantage, the graphical query viewer that Sybase ships with its server product line. Viewing SQL commands like `sp_helpdb` is much simpler when using a graphical utility.

Tip ✎

Links to download the excellent `ASEisql` and `sqsh` utilities can be found at http://www.edbarlow.com.

Sybase Recovery Procedure

The first step to recover from a database problem is diagnosing exactly what is wrong with the database. This section provides step-by-step directions for diagnosing and repairing server problems.

Here are a few important things to keep in mind when working on a Sybase server:

- It may take some time to have a support call returned. If you have a support contract, contact Sybase technical support early in the process. Sybase technical support will call you back within an hour of placing a priority 1 call.
- Take a deep breath before starting. Errors made during a disaster can have dramatic consequences.
- The online Sybase manuals, particularly the *Troubleshooting Guide*, can be very useful.

Step 1: Can You Connect to Your Server Using isql?

The first thing you should try after someone has reported a server problem is to connect to your database server using a utility such as `isql`.

If you succeed, go to Step 2. If you fail, go to Step 5.

Step 2: Run the Stored Procedure sp_who

Once you log in to the database using `isql`, you should run the stored procedure `sp_who`. This procedure tells you what

users are doing on your server and identifies blocked processes.

> **Tip ✎**
>
> If you have loaded my extended system stored procedure library, you can use `sp__whodo` instead of `sp_who`. `sp__whodo` shows you only active processes and, more importantly, clearly shows you blocked processes. `sp_who` output shows one row per connected user; this can be a lot of data on large systems.

If you have any blocked processes, go to Step 3. If you see any processes in `LOG SUSPEND` state, go to Step 4. If everything looks OK, you can also run the stored procedure `sp_lock` to do further diagnostics or `sp_helplogin` *`loginname`* to check whether the user is locked out because of security policy (such as too many password attempts).

If `sp_who` shows no obvious problems and you still think something is wrong with your server, go to Step 6, and check the server error log.

Step 3: Blocked Processes

If Step 2 revealed that some of your server processes are blocked, you must decide whether there is a problem that requires you to terminate one of them. If you find that a query is blocking other queries, you can kill it by terminating the client program or by using the Sybase T-SQL command `kill`.

Step 4: Log Suspend

If `sp_who` shows a process in `LOG SUSPEND` state, one of your databases has a full transaction log. You need to either manually truncate or extend the log, or kill the process that filled the log. Details on how to proceed are explained in the "Transaction Log " section earlier in this chapter.

Step 5: You Can't Connect Using isql

If you can't connect to your server using a utility such as `isql`, you should log in to the system that runs your server (using terminal services or `telnet`) and go to Step 6.

Step 6: Check the Sybase Server Error Log

When you have a major problem, it is always useful to check the server's error log. The error log is not always in the same location; it is essentially set in the run file that was used to start your database. But usually the server's error log is in the file *$SYBASE/$SYBASE_ASE/install/<SERVER>.log*. Review this file, which is in chronological order, for errors.

> **Tip ✎**
>
> Sybase errors usually have a number associated with them. If you have an error in your error log, you should look up the description of the error in the *Sybase Troubleshooting Guide*, which can be easily found online at http://sybooks.sybase.com. That guide has good explanations of each error and usually provides a step-by-step procedure to rectify the problem.

Step 7: Check Whether Your Server Is Running

Use Procedure 2 to see whether your server is running. If it is, but you cannot connect to it from a remote client (Step 5), go to Step 8. Otherwise, go to Step 9.

Step 8: Running Server but Can't Connect Remotely

Attempt to connect to the server from the system that runs that server. If you cannot, double-check that your environment variables are set correctly; look at the *SYBASE.sh* file that is in the *$SYBASE* directory. If you still cannot connect to the server, but the server is running on your system, follow the procedure to shut down the server, and then move to Step 9.

Step 9: Restart Your Server

A full explanation of how to restart your server can be found earlier in this chapter in the section "Procedure 1: How to Start Sybase." The following commands are a quick summary:

```
$ cd $SYBASE/$SYBASE_ASE/install
$./startserver -f run_MYSYBASESERVER
```

Look for the message `recovery is complete` near the end of a successful recovery process. If the server starts cleanly, you are done. If not, go to Step 10.

Step 10: Startup Failure

If your server does not start, you have a serious problem. Try the following steps; perhaps there is a simple solution to the problem.

Step 11: Contact Sybase Support Immediately

If you have a Sybase support contract, you should call the minute a problem on your production server appears to be serious. Sybase support is excellent and should not be considered optional if your data is critical. It can, however, take at least 30 minutes to get a return phone call even for a priority 1 production issue, so you should call them the minute you realize you have a serious issue.

Step 12: Able to Get Shared Memory?

Sybase uses shared memory extensively for interprocess communication and caching data pages for quick access. If Sybase is prevented from acquiring the minimum shared memory it needs, it displays an error message like the following:

```
00:2006/03/21 23:39:02.78 kernel  os_create_region: can't
allocate 2147479552 bytes
00:2006/03/21 23:39:02.80 kernel  kbcreate: couldn't create
kernel region.
00:2006/03/21 23:39:02.80 kernel  kistartup: could not create
shared memory
```

If your server does not start with a short error message like this, you are having a problem allocating your shared memory. There are a couple of explanations for this. The most obvious reason is that the memory has not been freed up from the last server shutdown. The fix for this is to remove a file *<SERVERNAME>.krg* that can be found in

$SYBASE/$SYBASE_ASE. Remove the *<SERVERNAME>.krg* file, and try to restart the server again.

If this does not work, find out if someone changed the memory configuration of the system or if you changed the amount of memory required by the server at startup. Figure out the amount of memory on your system, and compare it to the amount of memory that the server tried to allocate (2,147,479,552 bytes in the previous error message). If the server is trying to allocate too much memory, that value can be changed in the configuration file *<SERVERNAME>.cfg*.

Finally, Sybase requires that you set a few options in */etc/system*. These arguments define the maximum shared memory that the server can allocate. If you have not set these values (first-time install), you should check out the Sybase installation guide for your platform. If you have changed the amount of memory available on your system, the */etc/system* values may need to be updated.

You can view the shared memory being used using the `ipcs` command. Check the manpages for the correct usage of the command, but the output should look something like this:

```
------ Shared Memory Segments --------
shmid     owner     perms     bytes     nattch    status
129       sybase    600       11964416  1
2         sybase    666       1024      3
56        curtis    606       33334342  3
```

Here, the `curtis` process has also taken some of the shared memory for itself. By stopping this process, the shared memory should be freed up. If stopping the process does not free up the memory, it can be freed using the Unix command `ipcrm`. Again, check the Unix manpages for more information on this command on your operating system.

Tip ✎

If possible, make one change to the configuration options at a time. Sybase configuration options interact, some causing the system to need more memory, others less. By making one change at a time, the configuration option that prevents the system from restarting is known and can be adjusted accordingly.

Once you have corrected these problems, return to Step 1.

Step 13: Master Device Failure

If your server does not start because something is wrong with the master database, you will get the following error message during startup:

```
00:2006/03/22 00:53:48.44 kernel  kdconfig: unable to read
primary master device
00:2006/03/22 00:53:48.44 kernel  kiconfig: read of config block
failed
```

The primary master device contains the master database. This message indicates that the dataserver cannot read the master file. You should check the file permissions and ensure that the `sybase` account can access the file (you are starting the server as `sybase`, right?). The master device can be readily found by looking at the *run_FILE*. The Sybase `dataserver` executable identifies its master device with the –d flag.

If there are no problems accessing the master data device, the master database might be corrupted. This is an unlikely but fatal scenario; the server will not start if the master device is corrupt. Master device failure is a special case for the Sybase recovery procedure. If this occurs, you should

immediately contact Sybase support if you have a maintenance plan. You should also immediately go to the online Sybase troubleshooting guide and look at the section on recovering a failed master device from backups. It gives detailed step-by-step procedures for this scenario.

One main reason we do not store user tables in the master database is the critical nature of this database. Because we do not store transactional tables in master, this database should be stable and not change much. The master database is also generally a very small database.

The online *Sybase Troubleshooting Guide* provides recovery steps for corrupt master devices. Basically, you will be given a procedure to recover the system from a backup of master and a separate procedure to recreate the server from scratch. The first procedure is much superior because you will not need to recreate and reload any of your databases.

Step 14: Disk Device Failure

As mentioned earlier, disk failure is the most common problem you will encounter. You should use disk mirroring either at the operating system level (preferred) or at the database level to protect yourself from this problem. If your dataserver error log indicates the failure of a disk device, follow these procedures to check OS device problems, fix them, and maybe replace them. You should first check ownership and permissions on devices. The Sybase account should have read and write access to all disk devices and raw partitions that it needs. Check the specific file listed in the error log. You should also check the system log for device failure messages. Sometimes, because of a change in the system, the operating system might no longer “see” the device. There might be hardware failures or configuration errors that could cause the device not to

appear. Use the appropriate OS procedure to check the device's status, and review all error logs.

If the disk has gone bad, you have no choice but to rebuild the databases that it contained and reload them from backups. You need a configuration audit to do this.

Get a list of the databases that failed to load

The first thing you need is a list of the bad devices. Your server startup messages list the devices that did not start. It will also list the databases that did not start. Use the T-SQL statement in Example 17-6 to list databases that use a particular disk device.

Example 17-6. Locating databases that use a particular device

```
select sysdevices.name as DevName,
sysdatabases.name as DBName,
sysusages.size/512 as Size
from sysdatabases, sysusages, sysdevices
where
sysdevices.name="BadDeviceName" and
sysdevices.low <= sysusages.vstart and
sysdevices.high >= sysusages.vstart and
sysusages.dbid = sysdatabases.dbid

Example Output:

DevName                         DBName
Size
 ------------------------------ ------------------------------ ---
--------
 BusDev1                         BillingDB                       3
 BusDev1                         ClientDB                        2
```

Check your available free space

If you lose a 32 GB disk device, you should ensure that you have at least 32 GB of disk space that has been assigned to the server but not allocated to a database. You can use the `sp_helpdevice` procedure to do this. If you do not have

sufficient disk space available to replace the disk that went bad, you need to replace your disk and reassign it with the `disk init` command.

To drop a device, you can run the `sp_dropdevice` procedure.

If you have sufficient disk space, you can skip this step. If not, you will need to add a device to replace the bad disk. The `disk init` command is used for this. If the original disk's physical name is */dev/dsk/c0t2d1s0* with a `disk init` command of:

```
disk init
name="BusDev1, "
physname="/dev/dsk/c0t2d1s0, "
vdevno=6,size=2048
```

the new command using the replacement device */dev/dsk/c1t3d0s1* is:

```
disk init
name="BusDev1, "
physname="/dev/dsk/c1t3d0s1, "
vdevno=10,size=2048
```

Once the device has been recreated, all the databases that were using that device need to be restored.

Get database recreation information

At this point, you will need to get the information necessary to recreate your database. The commands in Example 17-7 will give you the file allocations used by this database.

Example 17-7. Database allocations

```
select sysdevices.name,
size as Blocks,
size/512 as Mbytes
from sysusages, sysdevices, sysdatabases
where sysdatabases.name = "dbname" and
sysusages.dbid = sysdatabases.dbid and
```

```
sysdevices.low <= sysusages.vstart and
sysdevices.high >= sysusages.vstart and
sysdevices.cntrltype = 0
order by vstart

Example Output:

name                    Blocks     Mbytes
----------------------- ---------- ---------
device1                 1536       3
logdev1                 1024       2
device3                 2048       4
```

Most database tools can reverse-engineer the exact SQL statement that you will need to recreate your databases. If you do not have a tool that does this, you can use the stored procedure `sp__revdb` from http://www.edbarlow.com.

Drop the broken database

If your database has a fatal error, you may need to drop the database using this T-SQL command:

```
$ drop database baddbname
```

It is possible that this command will fail. In that case, you must use `dbcc` to drop the database:

```
$ dbcc repairdb(dropdb,baddbname)
```

To verify the database has been dropped, run the stored procedure `sp_helpdb`.

Recreate the database

Using the database allocations determined earlier, recreate the database using the same allocations it had. Here is an example based upon the preceding example output:

```
create database baddbname
on device1 = 3
log on logdev1 = 2
```

```
alter database baddbname
on device3 = 4
```

Reload your database

Now reload the database using the most recent database and transaction dumps. First, apply the full database backup. For our example database, here is an example `load` from */dumps/baddbname.dmp*:

```
$ load database baddbname from '/dumps/baddbname.dmp'
```

After this completes, apply each transaction log starting with the oldest and finishing with the newest. The system will not allow transaction logs to be loaded out of order. In fact, if any of the logs are missing or corrupt, the rest of the logs cannot be applied. To load the transaction logs for the preceding example, enter the following command, repeating it for each transaction dump:

```
$ load transaction baddbnamefrom '/dumps/baddbname_trn.dmp'
```

For more information on how to load dumps and transaction logs, please refer to the section “Restoring from a Hot Backup,” earlier in this chapter.

Bring the database online

At this point, the database has been recreated, but the system will not bring it online until it is told to. The reason the system does this is that it has no way of knowing if there are any more transaction logs to process. The database should be brought online with the `online database` *baddbname* command.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 18. IBM DB2 Backup and Recovery

IBM DB2® Universal Database™ (DB2 UDB) is not a new player in the relational database management system (RDBMS) market. DB2's history and legacy begins with the concept of the RDBMS proposed by E. F. Cobb of IBM Research in 1970. Since then, IBM has developed a complete family of RDBMS software now called DB2 UDB. DB2 was first released in 1983. The database manager in OS/2®Extended Edition in 1987 was the first relational database on distributed systems. DB2 UDB is now available on Linux, Unix (AIX, Solaris, and HP-UX), and Windows platforms. The backup and restore utilities for DB2 UDB on these operating systems are virtually identical and platform-independent.

Tip ✎

This chapter was contributed by Jeff Richardson, Kondal Yennaram, and Kulvir S. Bhogal. Jeff has worked for IBM for 24 years, is a certified DB admin for DB2 UDB LUW, a martial arts instructor, a gardener, and a woodworker wannabe when he has free time. Kondal works for IBM as a Senior Software Specialist for DB2 UDBs on LUW. Kulvir Singh Bhogal works as a WebSphere consultant, implementing IBM's e-business strategies across the United States. Kulvir has nearly 100 technology patents pending with the U.S. Patent Office as well as 14 granted patents.

Backup and recovery have been integral parts of IBM DB2 UDB since its beginnings. DB2 UDB provides backup and restore commands, utilities, wizards, and APIs. Backup and recovery can be performed offline (cold) and online (hot),

providing true 24/7 availability of data. Backups can be scheduled to run automatically; if DB2 UDB detects that no backup is necessary, the backup does not execute.

This chapter describes, at a high level, the architecture of DB2 UDB and the components a DBA needs to perform backup and recovery, along with backup and restore examples. Detailed information concerning DB2 UDB architecture and the DB2 engine and its utilities can be found in the DB2 administration guides, *Command Reference*, the *Data Recovery and High Availability Guide and Reference*, and other manuals. These documents are available for download at http://www.ibm.com/ and can be browsed online at http://publib.boulder.ibm.com/infocenter/db2help/index.jsp.

DB2 Architecture

DB2 UDB is a distributed database system implemented in a client/server architecture. The DB2 UDB runtime client can be installed on a separate physical system from the server. Client code is installed by default with the server. Client and server code at the physical server is separated into different address spaces; that is, they do not share physical memory. Application code runs in the client process while server code runs in separate processes. Separate memory units are allocated for database managers (instances), databases, and applications.

The Power User's View

Before launching into this chapter, here are some key terms that should be familiar to power users.

Instance

The set of processes that manage data is called an *instance* in DB2. Each instance is a complete, independent environment. Each instance has separate security from other instances on the same machine (system); has its own databases and partitions, which other instances cannot access directly; controls what can be done to the data and manages system resources assigned to it; and contains all the database partitions defined for a given parallel database system. The DB2 instance process is started with the `db2start` command.

The DB2 instance process runs on the DB2 server and is responsible for enabling access to the specified database. Several processes are created when the instance process is started. These processes interact with each other, maintaining the database and connected applications. Of these processes, several background processes are prestarted; others start on an as-needed basis.

Databases

Each DB2 *database* is a collection of interrelated data, and each database includes a set of system catalog tables that describe the logical and physical structure of the objects in the database. Other aspects of the database structure are maintained in a database configuration parameter file and the recovery log.

Schemas

A *schema* is an identifier that qualifies tables and other database objects. A schema name is used as the first part of a two-part object name. For example, a schema named Smith might qualify a table named *smith.payroll*.

Tables

A relational database presents data as a collection of *tables*. Data in a table is arranged in columns and rows. The data in the table is logically related. Relationships can be defined between tables.

Views

A *view* provides a different way of looking at data in one or more tables; it is a named specification of a result table. A view has columns and rows just like a base table (a table created with the `create table` command). All views can be used just like base tables for data retrieval.

Indexes

An *index* is a set of keys, each pointing to rows in a table. An index allows efficient access when selecting a subset of rows in a table by creating a direct path to the data through pointers. The DB2 SQL Optimizer uses indexes to determine the most efficient (fastest) way to access data. The DB2 SQL Optimizer is a component of DB2's SQL compiler. It reviews statistics created with the `runstats` command, then chooses an access plan for a data manipulation language statement by selecting the one with the minimal estimated cost. The `runstats` command should be run after a database `reorg`, `restore`, or `recover` command.

DB2 engine dispatch units

Different operating systems dispatch tasks, threads, or processes. DB2 UDB server operations are performed by *engine dispatch units* (EDUs), implemented as processes or threads. DB2 uses the term EDU for consistency's sake across its varied platforms. Some DB2 background processes are started with the instance; others are

initialized when the database is activated by a connection. DB2 EDUs can be easily identified because they start with the string `db2`. For example, `db2gds`, `db2sysc`, and `db2wdog` are all DB2 EDUs.

The DBA's View

A DBA is typically concerned with the following architectural elements, as most pertain to the physical elements of the database.

Connecting to a DB2 database

To work with a DB2 database, the database manager instance's processes must be started, and your application or session must be connected to the database. To issue the following command, you must have `sysadm` or `sysctrl` authority:

```
% db2 connect to dbname user username using password
```

System catalog tables

The *system catalog tables* describe the logical and physical structure of the data and contain security information for database object access privileges. Catalog tables are created when the database is created and are updated during the course of normal operation. They cannot be explicitly created or dropped, but they can be queried and viewed.

Database partition

A database *partition* is part of a database that consists of its own data, indexes, configuration files, and transaction logs. A partitioned database is a database with two or more

partitions. Tables can then be placed in one or more database partitions using the partitioning feature.

Database partitions can reside on a single physical server, in which case the partitions are referred to as logical partitions. Alternatively, database partitions can also span multiple physical machines. (This is why DB2 database partitions are also known as nodes or database nodes.)

A database *partition group* is a set of one or more database partitions. You can create your own database partition group or use the default group.

A single-partition database, not surprisingly, has only one database partition. The partition still resides in a database partition group, but all data in the database is stored in the single partition. A partitioned database (or cluster) has two or more database partitions. Tables can be located in one or more database partitions. When a table is in a database partition group consisting of multiple partitions, some of its rows are stored in one partition, and other rows are stored in other partitions. You can create one or more database partitions on a physical system. The number of processors (CPUs) and amount of memory (RAM) installed in the system should be taken into consideration to meet your performance requirements. A database with multiple partitions is also known as a DB2 cluster.

In a multipartition database, the partitioning information is housed in the database node configuration file, *db2nodes.cfg*. The default location for this file is the database instance owner's home directory, specifically the */home/<instance_name>/sqllib* directory on Linux and Unix or *\Program Files\IBM\SQLLIB<INSTANCE_NAME>* on Windows. Each instance of DB2 has its own *db2nodes.cfg* file. Whenever a database is created under the instance, the database is partitioned based on the contents of the *db2nodes.cfg* file.

When a database is created, three partition groups are automatically created by default: the *ibmcatgroup*, *ibmtempgroup*, and *ibmdefaultgroup* partition groups. The *ibmcatgroup* partition group houses the DB2 catalog tablespace (such as *syscatspace*). This partition group consists of only one partition. The *ibmtempgroup* contains the system temporary tablespace (such as *tempspace1*), and the *ibmdefaultgroup* contains the user tablespace (such as *userspace1*). The *ibmtempgroup* and *ibmdefaultgroup* partition groups span all of the partitions of a database.

> **Tip ✎**
>
> To execute a command or SQL statement against all database partitions, you can use the `db2_all` command.

Since the `backup` and `restore` commands (described later in this chapter) operate against a single partition at a time, a multipartitioned database requires special attention during backup. DB2 requires that the catalog partition (that is, the partition that contains the catalog tables, which would be the partition that the `create database` command was executed on) be backed up or restored before any of the other partitions.

To do this, you can use the command:

```
C:> db2_all '<<+0< db2 backup db sample to backup_path'
```

The `<<+0<` specifies that only partition 0 is backed up.

After processing the catalog partition, other partitions can be backed up and restored in parallel. This parallel processing can be achieved using the command:

```
C:> db2_all '||<<-0< db2 backup db sample to backup_path'
```

where `||<<-0<` specifies the desire to run the backup of all partitions except for 0 in parallel.

Containers

A *container* is a physical storage device. It is identified by a directory name, a device name, or a filename. Each container can belong to only one tablespace. (Tablespaces are covered in the next section.) To find which containers are associated with a tablespace, run this command:

```
C:> db2 list tablespace containers for tablespace_num
```

In this example, *tablespace_num* is an integer representing one of the tablespace IDs returned from the `list tablespaces` command. For example, to see the containers for the *userspace1* tablespace (which has a tablespace ID of 2), run this command:

```
C:> db2 list tablespace containers for 2

            Tablespace Containers for Tablespace 2

 Container ID                            = 0
 Name                                    =
C:\DB2\NODE0000\SQL00002\SQLT0002.0
 Type                                    = Path
```

Tablespaces

A database is subdivided into *tablespaces* that are stored on one or more physical storage devices by defining containers on the different devices. Each DB2 database table is assigned to a tablespace, and multiple tables can reside in the same tablespace. Depending on the design and housing tablespace type chosen by the DBA who created the table, the DBA can have indexes associated to the table as well as large objects of the table (character large objects and binary large objects, for example) living in tablespaces other than that used to house the primary table data.

If a tablespace has multiple containers, DB2 spreads the data for a table housed in the tablespace across the containers in a uniform fashion. It is a common practice to use the fastest storage containers for a database's most frequently used tables and slower containers for less frequently used tables.

In DB2, tablespaces take on two different flavors: *system managed spaces* (SMS) and *database managed spaces* (DMS). A database can contain a combination of SMS and DMS tablespaces. Each container of an SMS tablespace is a directory in the file space of the operating system running DB2, whereas each container of a DMS tablespace can either be a fixed-size, preallocated file, or a physical device such as a disk. In practice, SMS tablespaces are typically used for small- to moderate-sized databases. DMS tablespaces are more difficult to set up but provide more flexibility. For example, with a DMS tablespace, a container can be added to a tablespace on the fly; you can't do that with an SMS tablespace. With a DMS tablespace, you can also split primary data, tables, indexes, and large objects into different tablespaces. With an SMS tablespace, all data for a table must be stored in the SMS tablespace.

When you first create a database, three tablespaces are created by default: *syscatspace*, *tempspace1*, and *userspace1*. The *syscatspace* tablespace contains system information about the objects that make up a database. The information is housed in DB2 system catalog tables and views. *tempspace1* is the tablespace used by DB2 when temporary tables must be dynamically created to handle such things as join operations. By default, when you create a table and do not explicitly specify a housing tablespace name, the table is created under *userspace1*.

It is important to know what tablespaces your database is made up of. As you will see later, with archive logging

enabled, you can back up at the tablespace granularity. To list the tablespaces of a given database, use this command (shown here for Windows):

```
C:> db2 list tablespaces
            Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
```

Large objects (LOBs)

DB2 also provides a specialized DMS tablespace called a *large tablespace*, also referred to as a *long tablespace*, which is specialized for the storage of large objects, whether binary, character, or graphic. LOB data will be included in regular backups.

Transaction logs

In DB2, *transaction logs* keep track of changes to a database, and they record how sets of changes are grouped as transactions. Think of transactions as a set of SQL statements that are executed as a single (that is, atomic) operation. The transaction logs keep track of whether a transaction is committed or rolled back. Transaction logs play a critical role in both crash recovery and rollforward recovery, both of which are discussed in the section "Recovery Types" later in this chapter. DB2 uses a technique known as *write ahead logging* in which transactions are logged while they occur, before any data is written to the database. Transaction logs are kept either in files or in raw devices.

You can classify transaction logfiles as either primary or secondary logfiles. Primary logfiles are allocated when the database is first connected to, or during database activation time (using the `activate database` command). The number of primary logfiles is defined by the `logprimary` parameter value of the database configuration file. Primary logfiles are created immediately; secondary logfiles are created dynamically on an as-needed basis by the database. Secondary logfiles are created when there is a need for more transaction log space (because all the primary logfiles are filled up, and there is no way an older primary logfile can be overwritten because it contains data from an active transaction). In such cases, the temporary need for more log space is fulfilled by creating secondary logfiles.

The `logsecond` parameter value of the database configuration file controls the maximum number of secondary logfiles that can be allocated. The size of logfiles is defined by the `logfilsiz` database configuration parameter; the unit for the value is 4 KB pages. Accordingly, if your database has a `logprimary` parameter

value of 2 and a `logfilsiz` parameter value of 300, your database has 2 primary logfiles with 300 4 KB pages.

If a log contains information about transactions that have not been committed or rolled back, or if a log contains information that has been committed but has not been externalized to the database disk, the log is considered *active*. On the other hand, if a log contains information about committed and subsequently externalized transactions (that is, transactions that have been persisted to disk), and the logs are located in the same disk as the active logs, these logs are *online archive logs*. Archive logs that have been moved from the active log directory to another directory or media are known as *offline archive logs*. You can move archive logs from the active log directory to another location either manually or automatically with the `userexit` parameter (in DB2 UDB V8.1 and previous) or the `logarchmeth1` and `logarchmeth2` parameters (in DB2 UDB V8.2 and later). The usage of these parameters is covered in the section "Managing archive logs," later in this chapter.

You can learn the path of DB2's files by looking at the values of the database configuration parameters. The database configuration also can tell you more information about your logging, such as the number of primary logfiles allowed (specified by the `logprimary` database configuration parameter) as well as the size of the logfiles (specified by `logfilsiz`). To see the value of these database configuration parameters, run this command:

```
C:> db2 get db cfg for sample | find /I "log"
```

In the Unix world, you can use `grep` in the same way to look for a particular database configuration parameter. For example, the following command helps pinpoint database configuration parameters related to the location of logfiles (the Windows `find` command would give similar output):

```
% db2 get db cfg for sample | grep -i log
Number of primary logfiles      (LOGPRIMARY) = 3
Number of secondary logfiles     (LOGSECOND) = 2
Changed path to logfiles        (NEWLOGPATH) =
Path to logfiles                             =
/home/db2inst1/NODE0000/SQL00002/SQLOGDIR/
Overflow log path           (OVERFLOWLOGPATH) =
Mirror log path               (MIRRORLOGPATH) =
```

To protect against media failure, keep the database logs on a different physical device from the database itself.

Managing archive logs

By default, when you create a DB2 database, it uses *circular logging*. In circular logging, when a log reaches its maximum size, the log wraps around (in a circle, hence the name circular logging) and overwrites earlier entries unless the logfile or files are still needed for crash recovery. In this case, you encounter a log-full condition. Therefore, the number of primary logfiles must be sufficient to allow at least one inactive logfile at all times.

Circular logging does not allow for rollforward recovery (explained in the section "Recovery Types" later in this chapter). For example, let's say we back up a database at time T0, and the database fails at time T1. To recover from the failure, we restore the database from the T0 backup. If circular logging is enabled (which means that rollforward recovery is not enabled), we would not be able to recover our delta changes from T0 to T1. In DB2 terms, circular logging therefore supports only crash and version recovery (also explained in "Recovery Types").

If a database is using circular logging, the database can be backed up only when no applications are connected to the database. This form of a backup is known as an *offline backup*. With circular logging, a backup operation also must be

performed on the entire database; you cannot perform tablespace-level backup.

If you also want to perform rollforward recovery, you must switch the database from circular logging to *archive logging*. When a database uses archive logging, it can be backed up when the database is online. (You can also perform offline backups if you choose.) In addition to being able to back up at the database level, archive logging allows you to back up at the tablespace level. This ability to pick and choose which tablespaces are backed up (and subsequently restored) during a backup operation allows you to create a more appropriate database backup plan in which more active (frequently changing) tablespaces can be backed up more often than less active (not so frequently changing) tablespaces.

In addition to allowing you to perform hot backups, archive logging also allows you to recover the database to the point of failure by applying transactions that were successfully committed since the database backup. Going back to our earlier example, let's say we back up a database at time T0, and the database fails at time T1. To recover from the failure, we restore our database from our backup at T0. If archive logging was enabled, we can recover our delta changes from T0 to T1. As you'll see later in this chapter, we are also not restricted to recovering our data all the way to T1 (the end of our logs). We can recover our data up until any point in time between T0 to T1.

There are two methods of enabling archive logging: keeping your archive logs in their original location as long as they are needed for recovery, or copying them to an alternate location to use during recovery.

If you want to keep the logs in their original location, you can set the value of the `logretain` parameter to `recovery`. This can be done with the `using logretain` option to the

db2 update db command. The following sequence of commands shows how to enable this parameter. A basic backup command is included because the database is immediately placed into a backup pending state when you enable and activate archive logging.

```
% db2 update db cfg for sample using logretain on
% db2 force applications all
% db2 terminate
% db2stop
% db2 backup db sample
%db2start
```

Setting logretain to recovery adds the complexity of having to worry about your disk filling up with logfiles as applications interact with your database and perform operations that cause your logfiles to grow. The more active your database is, the larger and quicker these logs grow. It is critical that you move old logfiles to an alternate location (ideally not the same location as your database) to prevent the disk from filling up. As mentioned earlier, in order to protect against media failure, you should keep database logs on a different physical device than the database itself. Also, for more frequently changing databases, online backups of the most active tablespaces should be performed more frequently. This way the database can be restored instead of rolled forward using archive logs, which results in longer downtime.

An alternative to leaving the archive logs in the database directory is to copy them to another location. The traditional method of doing this (prior to 8.2) was to set the userexit parameter to on. In 8.2, this parameter has been replaced with the logarchmeth1 and logarchmeth2 parameters.

If you are running DB2 8.2 or later, you should use the logarchmeth1 and logarchmeth2 parameters. These parameters are much easier to use than the userexit

parameter; the parameter name is short for *log archive method*. You can set these parameters in a few different ways:

Set logarchmeth1 to userexit

: Only `logarchmeth1` can be set to this value, and it is the equivalent to setting `userexit` to `on`. `userexit` is automatically updated for you.

Set logarchmeth1 to DISK

: If you set `logarchmeth1` to `DISK:/` *`path`* `/` *`directory`*, archive logs are automatically copied to the directory you specify. This method is much easier than creating a `userexit` script that accomplishes the same thing.

Set logarchmeth1 to TSM or VENDOR

: These arguments are meant for sending archive logs to a commercial backup product. TSM is short for Tivoli Storage Manager.

Optionally set logarchmeth2

: If you've set `logarchmeth1`, you can also set `logarchmeth2`. If you specify a value for both parameters, both values are used, and archive logs are archived twice. You can specify to archive to a second directory or to archive to TSM or another vendor.

If you are running 8.1 or would prefer to continue using `userexit` scripts in 8.2, you can also use the `userexit` parameter. When using this parameter, a user-supplied program (`userexit`) can automatically take a filled logfile and (if programmed to do so) copy the logfile to an offline archive. User exits also come into play when a database is rebuilt from a backup image. To account for the changes after the database backup, `userexit` is also responsible for retrieving the offline archive files that it stored externally.

In Unix systems, the `userexit` can be any executable program, such as a shell or Perl script, or a compiled program. It must be named *db2uext2* (with no extension) and must be stored in the *sqllib/bin* directory of a DB2 installation. In Windows, the user exit must be a compiled program named *db2uext2.exe* and needs to be stored in the *sqllib\bin* directory of the DB2 installation. DB2 provides sample `userexit` programs in the *sqllib/samples/c* directory.

If you want to use the `userexit` parameter, the following command sets this configuration parameter to `on` in an 8.1 (or prior) database:

```
C:> db2 update db cfg for sample using userexit on
```

If you want to use the `userexit` parameter in an 8.2 (or later) database, use this command. It automatically sets `userexit` to `on`.

```
C:> db2 update db cfg for sample using logarchmeth1 userexit
```

Warning ☠

When you change the values of `logretain`, `userexit`, `logarchmeth1`, or `logarchmeth2`, the database is put into a state known as *backup pending*. At this point, you must perform a backup (covered later in this chapter) of the database before you can use your database.

The backup, restore, rollforward, and recover Commands

In order to be able to recover your DB2 database after a serious failure, you are going to need to have a solid backup and recovery plan. This plan is built around the backup and restore commands, which then enable you to do different types of recoveries.

The backup Command

The DB2 `backup` command (as its name suggests) backs up a database (or tablespace) to one or more devices or directories on the DB2 server machine.

> **Tip ✎**
>
> To initiate the `backup` command, you must have the `sysadm`, `sysctrl`, or `sysmaint` database user authority.

If you enabled archive logging for the database you are trying to back up, you can perform an online backup (that is, you can back up the database when applications are connected to it). With archive logging, you also don’t have to back up the whole database. Rather, the backup process can be performed at the tablespace level. If you want to keep your enterprise database highly available, and you cannot afford large scheduled maintenance windows that render a database unreachable by consuming applications, you should consider using online backups.

When performing the `backup` command, you can specify the following:

- The database alias of the database you want to back up (required).
- The name of the devices or directories on which the backup files will be created. If no name is explicitly specified, the backup operation writes the backup image to the current working directory of the client computer.
- The name of the tablespaces you want to be backed up (available only if you have enabled archive logging).
- The `username` and `password` to use to perform the backup operation.
- Whether to perform the backup online or offline. Offline is the default mode of backup. Online backup is allowed only if you have enabled archive logging. It is important to note that at least one full offline database must have been performed after the `logretain` option was enabled for a database before an online database backup can take place.

> **Tip ✎**
>
> Some useful commands in your DBA toolbox include the `list applications` and `force applications all` commands. Respectively, these commands let you know what applications are connected to your database and allow you to kick off all the connected applications.

- Whether you want the backup to be either incremental or delta (see the section “Backup levels” later in this chapter).
- A `parallelism` value that allows you to specify the number of tablespaces the DB2 backup utility should

read in parallel when performing the backup. If no value is specified, DB2 automatically provides one for you.

- Whether you want DB2 to compress data during the backup, saving space on the backup device.

Here's an example of a `backup` command:

```
C:> db2 backup db sample userid db2admin using password to c:\backup
```

This command creates a full database backup of the database named `sample` to a directory named `C:\backup`. The database user `db2admin` with a password of `password` performs the `backup` command. As stated earlier, it is assumed that `db2admin` has the `sysadm`, `sysctrl`, or `sysmaint` database user authority. This backup command is performed offline (the default mode of operation). You can change it to an online backup by adding the word `online` just after the `password` phrase. (You must have archive logging enabled to do this.)

Here's another example:

```
C:> db2 backup db sample user db2admin using password tablespace(userspace1)
online to c:\backup
```

This command backs up only the *userspace1* tablespace of the *sample* database, showcasing the ability to back up at the tablespace level. This is highly useful when a database has a subset of tablespaces that change more frequently than the rest. It is important to note that you cannot back up temporary tablespaces using this command.

It is a good practice to back up related tablespaces together. Examples of such related tablespaces are tablespaces containing tables that have referential constraints between the tables. Another example of related tablespaces involves a typical topology in which the indexes

of a table and LOBs are housed in different tablespaces from the main table data.

Going back to our example, the tablespace backup is performed online, so it is assumed that archive logging has been enabled for the *sample* database.

After issuing such a `backup` command, you should receive a message similar to the following, telling you that the backup operation was successful:

```
Backup successful. The timestamp for this backup image is:
20061119181253.
```

Backup levels

When running the `backup` command, you can opt for a full, incremental, or delta backup. A *full* backup, as its name implies, contains all of the data of the database or tablespace being backed up. An incremental backup (also called a cumulative backup) makes a copy of all database data that has changed since the most recent successful full backup. On the other hand, a *delta* backup copies all database data that has changed since the most recent successful backup (full or other). Incremental and delta backups are therefore reliant on previous backup images and cannot be used by themselves to restore a database. This reliance on other, previously created, backup images makes it very important to save all the backup images needed to perform a database recovery.

Backup path and filenaming convention

DB2 automatically determines the pathname and filename for backup files. The generation of this name is far from random; it follows a strict naming convention. Dissecting *the name of a database backup path and file can let you know a lot about the DB2 backup that created the backup file.*

The following information is contained in the path and/or name of a backup file:

Database alias

The alias for the database.

Type of backup

0 for full database backup, 3 for tablespace(s) backup, 4 for a copy of rows loaded by the load utility.

UDB instance

The name of the UDB instance.

Database node number

For a single-partitioned database, NODE0000.

Catalog node number

For a single-partitioned database, CATN0000.

Timestamp of the backup

yyyy represents the year, *mm* represents the month (01 to 12), *dd* represents the day of the month (01 to 31), *hh* represents the hour (00 to 23), *mm* represents the minutes (00 to 59), and *ss* represents the seconds (00 to 59).

Sequence number

A three-digit number used as a file extension.

In Windows, this naming convention is represented by a pathname and a filename. In Unix, the naming convention is represented only by the filename. On Windows, a four-level subdirectory tree houses the backup file:

DB_alias.*Type**Inst_name*\NODE*nnnn*\CATNnnnn*yyyymmdd**hhmmss.Seq_num*

A backup filename in Windows might be:

```
SAMPLE.0\DB2INST\NODE0000\CATN0000\20060227\145655.001
```

For Linux and Unix, rather than dealing with a four-level subdirectory tree, all the information is built into the filename itself:

```
DB_alias.Type.Inst_name.NODEnnnn.CATNnnnn.timestamp.Seq_num
```

Using the same parameters as the Windows example, our Linux or Unix filename example would be:

```
SAMPLE.0.DB2INST.NODE0000.CATN0000.20060227.145655.001
```

Don’t Ignore Partial Backups

The backup software we were using at a U.K. insurance company reported a number of partial backups every night. (A partial backup is one in which one or more of the files were not backed up.) Since there were so many occurring on a nightly basis, they were not being investigated by the administrators. A business-critical DB2 database was performing full dumps to disk, and then incremental backups were being performed against those full dumps. Unfortunately, it was the nightly incremental backups of DB2 that were failing, which would have prevented us from recovering a business unit in a disaster situation. Checks were put in place to reduce the number of partial backups and investigate any that did occur. We dodged a bullet.

Hywel Matthews

Discovering the history of your backup operations

Backup, restore, and rollforward operations performed on a database are logged in a recovery history file named *db2rhist.asc*. Each database has its own recovery history file, located in the same directory as the database.

The `list history` command shows the backup, restore, and rollforward operations performed on a database. For example, to learn which backup and restore operations have been performed on the `sample` database, you can use this command:

```
C:> db2 list history backup all for db sample
```

The report generated by issuing this command contains a symbol indicating the operation performed (`B` for backup, `R` for restore). Here is a sample of the output from the `db2 list history` command:

```
                    List History File for sample

Number of matching file entries = 1


 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log
Backup ID
 -- --- ------------------ ---- --- ------------ ------------ ---
-----------
  B  D  20060409190953001   F    D  S0000000.LOG S0000000.LOG
 ------------------------------------------------------------------
------------
```

If the operation was a backup, a symbol indicates the type of backup:

```
F – Offline Backup
N – Online Backup
I - Incremental Offline Backup
O - Incremental Online Backup
D - Delta Offline Backup
E - Delta Online Backup
```

The backup report also tells you which tablespaces were backed up as well as the location of the backup image. This information can come in quite handy when trying to recover your database.

To learn which `rollforward` operations have been performed on the `sample` database, use this command:

```
C:> db2 list history rollforward all for db sample
```

The report generated by issuing this command contains an `F`, indicating that a `rollforward` operation was performed. The `type` column of the report indicates whether the rollforward was performed to the end of the logs (indicated by an `E`) or to a point in time (indicated by a `P`).

Automatic maintenance

DB2 has components with autonomic features and automated backup utilities. These components include database configuration parameters, the health monitor and its indicators, and a number of GUI interfaces (including the Health Center, the Web Health Center, the Task Center, and the Configure Automatic Maintenance wizard). If you are a power user, you can skip the GUIs and configure automatic backup using the appropriate command-line processor (CLP) commands. See the *IBM DB2 Universal Database System Monitor Guide and Reference* for detailed instructions. This section provides a high-level overview.

The DB2 health monitor runs on the database server. The health monitor provides the functionality required for automatic database backup and maintenance. The health monitor gathers information about the health of the system using health indicators. Using these indicators does not impose a performance penalty. They are not the same thing as snapshot monitor switches.

Health indicators exist at the instance, database, tablespace, and tablespace container levels. The DBA configures health indicators using the Health Center, the Web Health Center, the CLP, or APIs. DB2 monitors these indicators and can take actions identified by the DBA. These actions can include:

- Alerting the DBA of potential system problems via email or pager
- Executing a preconfigured action, such as increasing tablespace size or adding containers
- Logging alerts in the administration notification log

Health indicators fall into four categories:

Upperbounded threshold-based

This indicator type represents a statistic or percentage. For example, if you want to proactively monitor tablespace utilization, you can set the warning value to 70 and alarm value to 90 for the `ts.ts_util` indicator. This type of indicator has three valid states: normal, warning, and alarm.

Lowerbounded threshold-based

This indicator type also represents a statistic or percentage. For example, if you want to measure how much memory allocated for sorting is really being used, you can configure DB2 to notify you when the `db.max_sort_shrmem_util` indicator drops below 20 percent.

State-based

This indicator type represents a finite set of two or more distinct states of an object. One of the states is normal. All other states are considered abnormal, requiring attention. `db.db_backup_req`, for example, is the state-based indicator used to automate database backups.

Collection state-based

These indicators are database-level measurements that represent an aggregate state for one or more objects within the database. Collection state-based health

indicators also have two valid states: normal and attention. The db.tb_reorg_req (reorganization required) and db.tb_runstats_req (statistics collection required) indicators are examples of this type of indicator.

The db.db_backup_req health indicator is a state-based, database-level indicator. This indicator determines when a database backup is required based on either the time elapsed or the amount of data changed since the last backup. The auto_db_backup database configuration parameter must be set to on for automatic backups to occur. Automatic database backups can be either offline (cold) or online (hot).

The db.tb_reorg_req and db.tb_runstats_req indicators are set after a database is restored. Here is the recommended procedure for restoring a DB2 database:

1. Restore the database.
2. Reorg the database tables and indexes.
3. Collect statistics (runstats) concerning the tables and indexes.

Following this procedure helps enhance the performance of your applications. Reorg and runstats can be run at any time in offline (cold) or online (hot) mode. They do not require a database restore operation. However, you should include these steps in your database restore or recovery operating procedures.

A backup policy specifies automated maintenance behavior to ensure that the database is backed up regularly. The backup policy for a database is created automatically when the DB2 Health Monitor first runs. You can still perform manual backup operations when automatic maintenance is

configured. DB2 performs automatic backup operations only if they are required.

You define the time periods, or windows, for online and offline maintenance with the DB2 Health Center. DB2 then determines the need to perform a backup operation based on one or more of the following criteria:

- A full database backup has not been performed.
- The time elapsed since the last full backup is more than a specified number of hours.
- The transaction log space consumed since the last backup is more than a specified number of 4 KB pages (in archive logging mode only).

You configure the requested time or number of log pages between backups, the backup media, and the backup type (online or offline) using the Configure Automatic Maintenance wizard in the Control Center or Health Center. Automatic database backup can be enabled for either online (hot) or offline (cold) backup if the database is enabled for rollforward recovery (archive logging); otherwise, only offline backup is available. Automatic database backup supports disk, tape, Tivoli® Storage Manager (TSM), and vendor DLL media types.

If you select Backup to Disk, the automatic backup feature regularly deletes backup images from the directory specified in the Configure Automatic Maintenance wizard. Only the most recent backup image is guaranteed to be available at any given time. This directory should be kept exclusively for the automatic backup feature and not be used to store other backup images.

Offline backup, restore, reorg, and runstats restrict access to the database, as does online reorg and, to a certain extent, online restore. Offline database backups and table

and index reorganization are run in the offline maintenance time period that you define. These features run to completion even if they go beyond the time period specified. DB2's internal scheduling mechanism learns over time and estimates job completion times. If the offline time period you define is too small for a particular database backup or reorganization activity, the scheduler does not start the job the next time around. Instead, the health monitor notifies you if you need to increase the offline-maintenance time period.

You can set up automated backup, reorg, and runstats using the DB2 Health Center:

1. Start the Health Center GUI:
2. a. On Linux and Unix systems, log in as the DB2 instance, open a terminal session, and enter the command `db2hc`.

   b. On Windows systems, click Start→Programs→IBM DB2→Monitoring Tools→Health Center.
3. Expand the instance icon in the left frame, and right-click on the database icon. Three options are available: Configure Health Monitor Settings (where you set thresholds), Configure Automatic Maintenance, and Manage Utilities.
4. Click Configure Automatic Maintenance. The Configure Automatic Maintenance wizard appears. Six steps are listed in the left frame. Click Next twice to reach the Timing screen.
5. Set the time and duration for online and offline maintenance windows. Each maintenance window type has its own Change button. Click one of the buttons to set the values for the start time, duration, and days of

the week for this maintenance window type. Click Next when you are finished.

6. Set the notification email address. This page also provides troubleshooting assistance for notification problems.
7. Click Next to reach the Activities page, where you configure the settings for backup, reorg, and runstats. Highlight one of the maintenance activities, click the Automate checkbox, then click the Configure Settings button. There you will find the Backup Database (BACKUP) window that has three tabs: Backup criteria, Backup location, and Backup mode. The default mode is offline. Click Next to reach the final screen of the wizard, Summary. Click Finish to create and activate the automatic maintenance policy for the database.

The Health Center now shows two health indicators: Database backup required and Update statistics required. You should verify that automatic maintenance parameter settings for the database are enabled, substituting *db_name* with the name of the database for which you have created the automatic maintenance policy:

- On Unix or Linux, switch to the terminal session and enter this command:

```
% db2 get db cfg for
                                    db_name
                                    | grep –i auto_
```

- On Windows, open the DB2 command window on Windows by selecting Start→Programs→IBM DB2→Command Line Tools→Command Window. Enter the following command:

```
C:> db2 get db cfg for
                                    db_name
                                    | find /I "auto_"
```

If the database parameters are set to `off`, you can enable them with the following commands, which work on either platform (shown here on Windows):

```
C:> db2 connect to db_name
C:> db2 update db cfg using auto_maint on
C:> db2 update db cfg using auto_db_backup on
C:> db2 update db cfg using auto_tbl_maint on
C:> db2 update db cfg using auto_runstats on
C:> db2 update db cfg using auto_reorg on
C:>db2 connect reset
```

To reset the Database Backup Required Indicator, use this command:

```
C:> db2 backup dbdb_name
```

You have now automated the backup and maintenance of your database. Refer to the *IBM DB2 Universal Database System Monitor Guide and Reference* for information on setting other health indicators.

Using db2look

In addition to performing a backup, it's a good idea to run the `db2look` command against your database and save that output. `db2look` will display the statements necessary to reproduce the database objects of a database. The command below would create such an output for the sample database, and send it to *`path/file`*.

```
db2look -d sample -a -m -l -x -xd -f -o path/file
```

The output stored in *`path/file`* can come in very handy when recovering a DB2 database. It's a good idea to run this as soon as you create a database, or any time you make any structural changes to the database. Then place the output of this command someplace where you can easily retrieve it during a recovery operation.

Recovery Types

Before explaining the commands designed to recover a DB2 database, it's important to understand the different types of recoveries. DB2 UDB identifies the following three types of recoveries:

- Crash recovery
- Version recovery
- Rollforward recovery

Crash recovery

Crash recovery ensures that the database is brought to a consistent state after a software failure or power outage. Fortunately, crash recovery in DB2 is quite simple, and nothing special has to be done to a database to prepare it for crash recovery.

If you would rather have the database wait for you after a crash, you can turn off crash recovery by setting the `autorestart` database configuration parameter to `off`. If you change this setting, you must run the `restart` command to start the database after a crash:

```
% db2 restart db sample
```

The `restart` command effectively establishes a database connection. The database logs are then used to restore the database to a transaction-consistent state. Consequently, database changes made by committed transactions before the failure are made effective. On the other hand, rolled back transactions are rolled back in the database as are transactions that were in flight during the failure. At the end of the crash recovery, the DB2 database is restored to a transaction-consistent state.

Version recovery

If a database is damaged beyond the point where crash recovery can repair it (for example, the loss of a container), it must be restored from backup. Since this restores the database to a previous version, this is referred to as *version recovery*.

Rollforward recovery

A version recovery restores the database to the state it was in at the time the database backup was performed. Changes made to the database after a backup was taken are lost unless *rollforward recovery* is enabled prior to the failure and a `rollforward` command is issued after the restore (see the section "Performing a Rollforward Recovery" later in this chapter). Accordingly, in practice it's common to see a `restore` command used in tandem with a `rollforward` command.

DB2 8.2 introduced the `recover` command, which automatically performs a version recovery followed by a rollforward recovery.

The restore Command

A restore can be performed for a full database or a tablespace. It is important to note that a database restore must be performed offline whereas a tablespace restore (of any tablespace other than the one containing the system catalog tables) can be performed either online or offline. As mentioned earlier, an online tablespace restore is more desirable for enterprises wanting to minimize (if not eliminate) a database's downtime during the recovery of a database.

> **Tip ✎**
>
> To execute the `restore` command, you must have `sysadm`, `sysctrl`, or `sysmaint` database authority if you intend to restore to an existing database. If you plan on doing a redirected restore (covered later in this chapter), you need either `sysadm` or `sysctrl` authority.

The following information can be specified with the `restore` command:

- The name of the database backup that is being used as the source for the restore (required).
- The devices or directories where the database backup(s) are stored.
- If multiple backups are located on the database backup device or in the directory where the database backup(s) are stored, a timestamp can narrow down which backup to use for the restore operation. If you recall, the timestamp is part of the naming convention of the database backup.
- Whether the restore should be directed to a different database than the one from which it was backed up. This is a concept known as *redirected restore*, which is covered later in this chapter.

By default, if archive logging was enabled for a database, when you issue the `restore` command, the database is left in a *rollforward pending* state. The database or tablespace in a rollforward pending state cannot be used until it is brought out of this state by applying the `rollforward` command, as described later in this chapter. The `rollforward` command can account for the transactions that were committed after the backup occurred.

Warning ☠

A `restore` command can be issued with the clause `without rolling forward`. Doing so makes a database usable immediately after the `restore` command executes. However, remember that all the transactions committed after the database backup occurred are unaccounted for.

The rollforward Command

Once a database has been restored with the restore command, it can be rolled forward with the `db2 rollforward` command. The command allows you to roll forward to the last transaction log found or to a particular point in time prior to the last transaction.

Tip ✎

To execute the `rollforward` command, you must have `sysadm`, `sysctrl`, or `sysmaint` database authority if you intend to restore to an existing database. If you plan on doing a redirected restore (covered later in this chapter), you need either `sysadm` or `sysctrl` authority.

The recover Command

DB2 version 8.2 introduced the `recover` command. This command combines the `restore` and `rollforward` commands into a single command; thus it has some options from both. It has a few caveats, though:

- It can be used only to both restore and roll forward of an entire database.
- It cannot perform just the restore or just the rollforward. The `without rolling forward` option is not available. If you want to restore without rolling forward, you should use the `restore` command.
- It does not support tablespace-level recoveries.
- It does not support incremental recoveries.
- The `buffer`, `dlreport`, `without datalink`, `parallelism`, and `without prompting` options to the `restore` command are not available in the `recover` command.
- You do not tell it which backups to use. You tell it what time you want to recover the database to, and it automatically determines which backup is prior to that time. (Remember, it must restore from a backup that is earlier than the point in time you've chosen, and then rollforward to that point in time.)

> **Tip ✎**
>
> To execute the `recover` command, you must have `sysadm`, `sysctrl`, or `sysmaint` database authority if you intend to restore to an existing database. If you plan on doing a redirected restore (covered later in this chapter), you need either `sysadm` or `sysctrl` authority.

To recover a database using the `recover` command, use this command:

```
C:> db2 recover db DATABASE_NAME_OR_ALIAS
```

This automatically uses the best available backup image as determined from the recovery history file (described in the

next section) and recovers the database to the last transaction recorded in the transaction log.

To recover to a particular point in time rather than to the end of logs, you can use this syntax:

```
recover DB DATABASE_NAME_OR_ALIAS TO POINT_IN_TIME
```

To recover the database `sample` to the point in time 2006-04-10-00.16.52, use the following command:

```
C:> db2 recover db sample to 2006-04-10-00.16.52
```

The point in time is specified in local time, not UTC time. With the `recover` commands shown so far, the point in time must be contained in the current history file. If you need to restore to a point in time not contained in the current history file, explicitly point to an external history file. For example, to point to a history file archived in the directory */home/user/archives/*, use this command:

```
C:> db2 recover db sample to 2006-04-10-00.16.52 using history file
(/home/user/archives/db2rhist.asc)
```

Recovering Your Database

As mentioned previously, a DB2 database is recovered in two primary steps: a version recovery, in which the database is restored from backup, and a rollforward recovery, in which the transaction log replays transactions that have occurred since the last backup. This section first covers the two different scenarios in which you would perform a version recovery and then explains how to do a rollforward recovery against a restored database.

Performing an In-Place Version Recovery

If you are restoring a database in the location where it was originally backed up, we'll call it an *in-place* restore, to differentiate it from a *redirected restore*, in which you recover to another location. (Redirected restores are covered in the next section.)

Step 1: Gather your database backups

In order to restore a database or one of its tablespaces, we need to have access to our backups. This may sound obvious, but it is not so trivial when dealing with incremental and delta backups (see the section "Backup levels" earlier in this chapter). In short, we need to have all of the database backups necessary to restore our database/tablespace(s). If the files were moved to a different machine or network, gather the backup files in preparation for the restore operation. If you can't find all your backup files, you might want to consider digging into the recovery history file of your database (see the section "Discovering the history of your backup operations" earlier in this chapter). The recovery history file can tell you where

the database backups were created and the names of their files if they were stored on disk. If possible, put the backup files needed for your restore into a single directory. For our example, we'll use the directory *C:\backups*.

Step 2: Make sure the containers that existed during your backup are still around

If you do not intend to perform a redirected restore of your database and want the database backup to be restored into the database it came from, it is important that the containers that existed at the time when the backup image was made are still around. This is commonly not the case if you have to restore your database to a new machine after the original machine failed. If the tablespace containers that existed when your database backup was performed don't exist in the location you are trying to restore, you get an error during the recovery operation. To avoid such a problem, you can use a redirected restore (as described later in this chapter).

Step 3: Issue the restore or recover database command

We are finally ready to issue the `restore database` command. If you're running a version prior to 8.2, use the following syntax to restore a database:

```
C:> restore DB database_name_or_alias from backup_location taken at
yyyymmddhhmmss
```

For example, the following command specifies that we want to restore our database named `sample` from a backup identified by the timestamp `20060227145655`:

```
C:> db2 restore db sample from c:\backups taken at 20060227145655 replace
existing
```

As mentioned earlier, the timestamp of a backup is part of its filename. The `replace existing` clause states that existing data (if any) is deleted and replaced by the content of our backup.

> **Warning ☠**
>
> You can also add the `without rolling forward` clause at the end of the restore. Using this clause causes you to lose any transactions that occurred after the backup, but it makes your database usable immediately after the `restore` command executes. Therefore, while this clause brings the database online quicker, it will most likely result in a loss of data.

You can also restore just a tablespace. Tablespaces can be restored online or offline (except for a tablespace containing the system catalog tables, which must be restored offline). Tablespaces can be restored from full-fledged database backups or from backups that were performed at the tablespace granularity. You can use the following syntax to restore tablespace(s) of a database:

```
restore database database_name_or_alias tablespace
(tablespace_name, other_table_space_name,...) online from backup_location taken
at yyyymmddhhmmss replace existing
```

For example, the following command performs an online restore of the tablespace named `userspace1` of the `sample` database from a backup identified by the timestamp `20060227145655` and located in *C:\backups*.

```
C:> db2 restore db sample tablespace(userspace1) online from c:\backups taken at
20060227145655 replace existing
```

If you're running 8.2 or later, and you want to recover the entire database, you have the option of using the recover command instead:

```
C:> db2 recover db sample to isotime
```

This command automatically selects a backup that is prior to *isotime*, restores it, then rolls forward to the most recent transaction that it can find prior to `isotime`.

Step 4: Perform rollforward recovery

> **Tip ✎**
>
> If you use the `recover` command in 8.2, you can skip this step.

If you did not include the `without rolling forward` clause when issuing your `restore` command, the database is left in a rollforward pending state after the `restore` command completes. Tablespace restoration always places the restored tablespace(s) in the rollforward pending state. The database cannot be used until it is brought out of this state with the `rollforward` command. See the section "Performing a Rollforward Recovery" later in this chapter.

Step 5: Reorganize the data and collect statistics

This step is covered in detail in the section "Reorganizing Data and Collecting Statistics" later in this chapter.

Performing a Redirected Version Recovery

The filesystem paths used by the database are stored in the backup image. If you attempt to restore a database backup to a system that does not have the filesystems and physical devices that the database backup is expecting, you will receive an error during the database restoration operation.

To avoid this, you can use the redirected restore operation. It consists of three steps.

Step 1: Restore the database backup and specify the redirect option

Assume that you have a valid backup that you want to recover. Let's use our earlier example from the `restore` command procedure, but this time perform a redirected restore:

```
C:> db2 restore db sample from c:\backups taken at 20060227145655 redirect
```

By specifying the `redirect` option, DB2 actually pauses so that you can define the appropriate tablespace containers for your target database.

> **Tip ✎**
>
> The `recover` command does not yet support redirected restores, so you need to use the `restore` and `rollforward` commands.

Step 2: Define appropriate tablespace containers for the target database

At this point, we need to define the appropriate tablespace containers for our target database. The question is, how do we know what tablespace containers we need? This is where we can use the `list tablespaces show detail` command or the `list tablespace containers for` *`tablespace_num`* command, where *`tablespace_num`* is an integer representing one of the entries returned from issuing the `list tablespaces` command.

Of course, you cannot learn about tablespace container information from the source database if the source database has failed. Accordingly, you should get information about the tablespaces in the source database being backed up and put it away for safekeeping when needed for a redirected restore operation. You can use the output of the `db2look` command to provide this information (see the section "Using db2look" earlier in this chapter).

Next, define tablespace containers for the tablespaces that are associated with the database backup being restored. For example, the following commands create new tablespace containers for the `syscatspace`, `tempspace1`, and `userspace1` tablespaces. In this case, we use directory names relative to the database directory, not absolute pathnames.

```
C:> db2 "set tablespace containers for 0 using (path 'tbsp0cont1')"
C:> db2 "set tablespace containers for 1 using (path 'tbsp1cont1')"
C:>db2 "set tablespace containers for 2 using (path 'tbsp2cont1')"
```

Step 3: Continue the redirected restore operation

In Step 1, we were able to pause the `restore db` operation by specifying the `redirect` option. Now that we have defined the needed containers in Step 2, we can proceed with restoring the database:

```
C:> db2 restore db sample continue
```

Step 4: Perform rollforward recovery

As was the case with our in-place restore, if you are interested in accounting for transactions that occurred after our backup and after archive logging was enabled, you need to perform a rollforward recovery by issuing the `rollforward` command described in the following section "Performing a Rollforward Recovery."

Step 5: Reorganize the data and collect statistics

This step is covered in detail in the section "Reorganizing Data and Collecting Statistics" later in this chapter.

Performing a Rollforward Recovery

> **Tip ✎**
>
> If you used the `recover` command, you can skip this section, because the rollforward has already happened.

As mentioned earlier, a database `restore` command takes the database only to the state it was in when the `backup` command was performed on it. It is very likely that changes to the database occurred after your last available backup, and you would like those changes reapplied to the database. As mentioned earlier, if archive logging was enabled for a database, the database/tablespace(s) you restored are left in a rollforward pending state after a restore. A database or tablespace(s) in the rollforward pending state cannot be used until it is brought out of this state by applying the `rollforward` command to the database/tablespace(s).

A *rollforward recovery* allows DB2 to use its transaction logs to restore a database or a tablespace to the state it was in after the backup. This point in time can be either all the way to the end of the archive logs or some time before the end of the logs. However, the stipulation is that the point in time (PIT) must be at least the minimum point in time allowed for the tablespace(s). This minimum PIT ensures that the tablespace and logs are consistent with the system catalogs. To illustrate the concept of a minimum PIT, suppose you have a tablespace named *userspace1* that you

backed up at time T0. At time T1, you decide to create a new table in tablespace *userspace1*. By doing so, you effectively set the minimum PIT to T1 because if after T1 you tried rolling forward to some point between T0 and T1, the system catalogs would clash with the rollforward recovery. The rollforward recovery would not account for the new table. Accordingly, in order to avoid such synchronization issues with the system catalogs, DB2 forces a rollforward operation to at least the minimum PIT.

In this example, the minimum PIT was updated via a table creation. Other factors can affect the minimum PIT. Whenever DDL statements are run against the tablespace or against tables constituting the tablespace, the minimum PIT is affected. Since you cannot perform a rollforward operation on a tablespace using a value prior to the minimum PIT, you might be asking yourself how to determine the minimum PIT for a tablespace. The minimum PIT can be obtained using these commands:

```
C:> db2 connect to sample
C:> db2 create table test like staff in userspace1
C:>db2 list tablespaces show detail
```

The *userspace1* information now looks like this:

```
Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 408
 Useable pages                        = 408
 Used pages                           = 408
 Free pages                           = Not applicable
 High water mark (pages)              = Not applicable
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 1
```

```
 Minimum recovery time                = 2006-03-01-
13.23.19.232911
```

After performing a rollforward operation, committed transactions in the log are restored into the database; incomplete (that is, uncommitted) transactions in the log are rolled back. At the end of our rollforward recovery, we have a database in a transaction-consistent state, ready for use by applications.

As you can see, for a rollforward recovery to work, it is paramount that the logs of a database remain pristine and intact. As such, you should consider using RAID arrays and other precautionary hardware to ensure the safety of the DB2 logs.

> **Tip ✎**
>
> To perform the `rollforward` command, you must have the `sysadm`, `sysctrl`, or `sysmaint` database authority.

As with the DB2 `restore` command, the `rollforward` command can be applied in `offline` mode only if you are restoring at the database level. If you are restoring at the tablespace level, you can rollforward online. As with the DB2 `restore` command, an online rollforward operation cannot be performed on the system catalog tablespace (*syscatspace*). The *syscatspace* must be treated with uniqueness because this tablespace must be rolled forward all the way to the end of the logs.

> **Tip ✎**
>
> After a tablespace PIT `rollforward` operation is completed, DB2 puts the tablespace in the backup pending state. At this point you must perform a backup of the database before you use it.

The following sections describe the procedure to roll forward a database/tablespace(s).

Step 1: Gather your logfiles

In order for rollforward recovery to work, you need to have all your logfiles. The `rollforward` command assumes that the logfiles are in the log directory specified in the `logpath` configuration parameter; alternatively, you can specify the `rollforward` command with an `overflow log path` log directory. The `overflow log path` directory might be where a user exit saves archived logs or where the `logarchmeth1` parameter, available in 8.2 and later, copies them. If you are recuperating from a disk failure, move the archived logs to the `overflow log path` directory so the `rollforward` command can see the archived logs.

Step 2: Determine the minimum PIT

If you plan to roll forward tablespaces to a PIT prior to the end of your logs, this PIT must be equal to or greater than the minimum PIT for your tablespaces. As mentioned earlier, you can examine the minimum PIT with this command:

```
C:> db2 list tablespaces show detail
```

Step 3: Issue the rollforward command

Let’s cover a few examples of the `rollforward` command to exemplify how you might use it.

```
C:> db2 rollforward db sample user db2admin using password to 2006-03-01-
13.23.19.232911 and stop
```

This command rolls the `sample` database forward. In this example, the rollforward operation is carried out under the username `db2admin` with a password of `password`. The timestamp, `2006-03-01-13.23.19.232911`, specifies the point that the `rollforward` should stop. As you can imagine, this flexibility of PIT recovery can help if a database is corrupted by some rogue or even trusted application. Simply `recover` your database and then `rollforward` to before things went bad. The `and stop` clause in the example command tells DB2 to take the database out of the `rollforward pending` state so you can start using it.

If you want to roll forward all the way to the end of our logs, use this command:

```
C:> db2 rollforward db sample user db2admin using password to end of logs and
stop
```

You may have noticed that we did not try to perform the `rollforward` operations using the `online` keyword. This is because you cannot perform `rollforward` operations online at the database level. You can do this at the tablespace level, though, as shown in the following example:

```
C:> db2 rollforward db sample user db2admin using password to end of logs and
stop tablespace(userspace1) online
```

This command rolls forward the sample database’s tablespace named *userspace1*. The command is issued following the `restore` command for the *userspace1* tablespace. The rollforward operation is carried out under the username `db2admin` with a password of `password`. An

online rollforward operation cannot be performed on the system catalog tablespace (*syscatspace*). As done earlier, the `and stop` clause of our example command tells DB2 to take our database out of the `rollforward pending` state so that we can start using the database.

Step 4: Set constraints (if necessary)

Special attention also needs to be paid when rolling forward tablespaces with tables that have referential integrity relationships with tables housed in other tablespaces. When you roll forward such a tablespace, DB2 leaves the table in the check pending state. This state prevents the table from being used until you check its constraints. You can restore a table from the check pending state to a normal state by executing a `set constraints` statement.

When issuing the `set constraints` command, it is a good idea to specify an exception table into which rows violating the defined constraints are placed. The following command removes the table *test* from the check pending state. But before doing so, the contents of the table are checked against defined constraints for the table. Violators of the constraints are placed in the `badtest` table. To finish things up, the table is returned to the normal state. Future updates to the table are performed only if the updates do not violate the constraints of the table.

```
C:> db2 set constraints for test immediate checked for exception in test use
badtest
```

Step 5: Perform a database backup (if necessary)

If you performed a rollforward operation at the tablespace level in Step 3, your tablespaces are in a backup pending

state. You must perform a database backup before using the tablespaces.

Step 6: Reorganize data and collect statistics

This step is covered in detail in the next section.

Reorganizing Data and Collecting Statistics

The following commands optimize access to your data after a recovery. This example allows applications to access the data while the maintenance is performed.

```
C:> db2 connect to sample
C:>db2 reorgchk update statistics on table all > reorgchk.txt
```

The `reorgchk` command calls the `runstats` command and gathers a new set of statistics for the database. It returns a lot of data, so we redirect the output to a file for further evaluation. The last field of the `reorgchk` command's output identifies which tables need to be reorganized. Those tables are identified with an asterisk in one of several columns in the field. Assume the following output was returned from the `reorgchk` command. The `department` table would not need to be reorganized. It would be a good idea, however, to reorganize the data in the `employee` table.

```
SCHEMA    NAME         CARD    OV    NP    FP ACTBLK    TSIZE  F1
F2  F3 REORG
------------------------------------------------------------------
--------------
Table: ADMINISTRATOR.DEPARTMENT
ADMINIST> DEPARTMENT      9     0     1     1      -      549   0
- 100 ---
Table: ADMINISTRATOR.EMPLOYEE
ADMINIST> EMPLOYEE       32     0     2     2      -     2784   0
69 100 -*-
```

You reorganize the `employee` table with the following commands, allowing access to the data during the reorganization. Note that you need to fully qualify the table with the schema name in the `runstats` command.

```
C:> db2 reorg table employee allow read access
C:>db2 runstats on table administrator.employee allow write access
```

Hopefully, this chapter about how to back up and recover your DB2 database has been helpful. You should also make sure you are familiar with the DB2 manuals on the subject, especially the *Data Recovery and High Availability Guide and Reference* for your particular version.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 19. SQL Server

Easily the most popular database on Windows systems today, SQL Server originated in 1989 as a joint project between Microsoft, Sybase, and Ashton-Tate. It was essentially an OS/2 port of Sybase's SQL Server on Unix. SQL Server 4.2, the first version on Windows, shipped in 1992.

> **Tip ✎**
>
> This chapter was contributed by Scott Harris. Scott is a brand new father who's wondering how old his son Zach needs to be before he starts teaching him how to write Perl.

The partnership between Sybase and Microsoft began to break down due to differences of opinion on NT-specific code. Microsoft understandably wanted to customize the code for NT, and Sybase understandably wanted to maintain as generic a codebase as possible. SQL Server is now entirely produced by Microsoft and is completely different from the product produced by Sybase for the Windows platform. (Sybase changed the name of their product to Sybase Adaptive Server to avoid confusion with SQL Server on Windows.) Major versions include SQL Server 7.0, the first completely GUI-based database server, and SQL Server 2000, which was the first database written for the Intel IA64 architecture.

SQL Server code has changed significantly from its Sybase origins. As evidence of that, it's been 12 years since SQL Server has carried Sybase copyright notices. Nevertheless, with concepts like the *master* database and commands like

`dump database`, Sybase users may occasionally feel a sense of déjà vu.

SQL Server 2005, the currently shipping version at this writing, has grown beyond the standard basic database package that includes only the database engine. SQL Server 2005 also offers a wealth of additional features such as services for analysis, data integration, notification, and reporting as well as the service broker. Combined, these elements make up a complete relational database system that can be used for simple tasks such as a database-driven web application or for more advanced needs such as data mining, complex business intelligence gathering, specialized reporting and notification, and a host of additional needs.

> **Tip ✎**
>
> When referring to specific versions, this chapter often uses just the version numbers. For example, *2000* refers to SQL Server 2000, and *2005* refers to SQL Server 2005. When referring to both SQL Server 2000 and 2005, I use the term *SQL Server*.

Overview of SQL Server

As with any piece of complex software, SQL Server comes in multiple configurations. Based on a particular organization’s needs, the appropriate edition can be selected to minimize both cost and extraneous components. SQL Server 2005 comes in four primary editions and two specialized editions for a total of six from which to choose. They include the workgroup edition, standard edition, enterprise edition, developer edition, mobile edition, and

express edition. Consult Microsoft documentation for the differences between the editions.

Connecting to and Administering SQL Server

SQL Server 2005 can be administered from the GUI or from the command line. Versions up to 2000 are administered using the Enterprise Manager, and 2005 is administered with the SQL Server Management Studio (SSMS). This tool combines many of the previous individual tools, such as the Query Analyzer and the Analysis Manger, into one. For those who prefer the command line, a new program called `SQLCMD. EXE` is provided in 2005. This tool replaces `OSQL.EXE` and `ISQL.exe` from previous versions. In addition, the bulk data import and export program, `BCP.EXE`, is included to facilitate faster exports and imports of data to and from SQL Server databases.

SQL Server Authentication

Because of its integration with Active Directory, there are multiple ways to authenticate database users.

User authentication

SQL Server offers user authentication either through Windows Active Directory integration or from SQL Server user accounts. There is greater control over access when using Active Directory-integrated authentication, either through permissions on specific user accounts or through group permissions.

Service authentication

The actual SQL Server service has two different types of authentication available that allow the service to start:

Domain accounts

: Consider starting the service with a domain account if you manage multiple servers or your installation needs access to network resources. If using multiple servers, security administration can be simplified through domain policies, allowing preferences to propagate to specified or all servers. If access to network resources is needed, your installation needs a domain account to access shared network drives or other network resources.

Local system account

: A local system account should be used if your installation does not need access to network resources or to be authenticated against the domain. This is useful to isolate the server from the rest of the domain, which is normally done for security considerations, such as a server on an insecure network.

The Power User's View

Power users typically understand these elements of SQL Server architecture. They may use them when conversing with database administrators and system administrators.

Instance

An *instance* is a copy of SQL Server running on a computer. With SQL Server, you can have multiple instances running simultaneously. With versions prior to 2000, you can have only one instance running per machine.

Databases

A *database* in SQL Server is a collection of tables, indexes, and other objects that store data in a structured, relational format. An individual instance of SQL Server can support many different databases. Each database can store data that is accessed by the other, or they can be completely independent. SQL Server databases are divided into two generic categories, system databases and user databases. *System databases* hold information pertaining to the running system as well as configuration information for all databases. The *master database* is the best example of a system database. *User databases* store user data, hence the name. A database can have many states depending on its current condition; these include online, offline, restoring, recovering, recovery pending, suspect, and emergency.

> **Tip ✎**
>
> SQL Server 2005 is very robust. While not generally practiced, it is possible to have up to 32,767 databases per instance of SQL Server 2005.

System databases

System databases are the databases that SQL Server uses to manage and configure the running system. By default, the following system databases are installed:

master

: Contains server-specific configuation information as well as configuration information about all databases

model

: Contains a sample database on which you can base your databases

tempdb

: Provides a temporary area for processing queries and other tasks

msdb

: Used by the SQL Server Agent services for handling alerts, notifications and scheduled tasks

distribution

Used by the Replication Services when a server is configured as a publisher or distributor

> **Warning ☠**
>
> Never create user objects such as tables, stored procedures, or triggers in the *master* database. The *master* database is used only to house system-level information used by the instance.

User databases

User databases are the databases specifically for nonsystem information. This is the location where users store all their information.

Viewing information about databases

Configuration information about the database can be obtained using system stored procedures (defined later in this section) via Transact-SQL. To view information relating to a particular database, issue the following Transact-SQL command:

```
sp_helpdb dbname
```

This dumps the configuration information for a particular database. For our *Inventory* database, the output would look like this:

```
name        db_size  owner            dbid   created     status  compatibility_level
Inventory 3.00 MB  Administrator  7     Mar 21 2006  Status=ONLINE 90
                                        Updateability=READ_WRITE,
                                        UserAccess=MULTI_USER,
                                        Recovery=FULL,
                                        Version=611,
                                        Collation=SQL_Latin1_General_CP1_CI_AS,
                                        SQLSortOrder=52,
                                        IsAutoCreateStatistics,
                                        IsAutoUpdateStatistics
name           fileid   filename
Inventory     1    C:\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\Inventory.mdf
Inventory_log 2    C:\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\Inventory_log.ldf

filegroup     size       maxsize         growth       usage
PRIMARY       2048 KB    Unlimited       1024 KB      data only
NULL          1024 KB    2147483648 KB   10%          log only
```

This information is, of course, also available using the Management Studio in 2005 or the Enterprise Manager in 2000.

Tables

A *table* is a database object where data is stored and organized into rows and columns. As in other versions, 2005 can have up to 1,024 columns in any one table. Each column is configured to store a certain type of data—for example, character data, numeric data, or binary data.

System tables

In addition to user tables as defined previously, SQL Server stores the configuration of the server in *system tables*. These tables cannot be updated or queried directly by the user, but they can be queried through a dedicated administrator connect (DAC) or through the catalog views. The schema of these tables typically change from version of version, so take note if you are trying to install an older application that accesses a newer version of SQL Server.

Temporary tables

Temporary tables are just like regular user tables except they are created in the *tempdb* and are only for short-term use. There are two types of temporary tables to choose from based on how long you need the temporary tables to stay around: local and global. *Local* temporary tables are available only to the user that created them and only during the same connected session to the SQL Server instance. These tables are deleted after the session disconnects. *Global* temporary tables are available to any connected session or user. These are deleted after all sessions referencing the table are disconnected.

Index

Indexes are created on tables or views in order to make lookups on the data faster. Indexes are created from one or more columns and form a key. By using indexes, much less data needs to be traversed in order to find the data that you need. This can greatly speed up the operation of your application.

Partitioned tables

A *partitioned table* is a special type of table that has its data divided across more than one filegroup in the database. (Filegroups are defined in the section "The DBA's View" later in this chapter.) By doing this, larger tables become more manageable, and the data within them can be accessed faster. This happens because the data within the partitioned table can be accessed or manipulated in subsets, which means that only part of the table needs to be read to find the result.

Partitioned indexes

Partitioned indexes are indexes spread over more than one filegroup. These, along with partitioned tables, make indexes more manageable and speed up access.

Stored Procedures

Stored procedures are precompiled Transact SQL statements that are stored in the database. Stored procedures provide an excellent way to execute code that would otherwise be executed outside the database. They offer a great performance benefit because you save the overhead of having to transfer the data to some outside program or client before performing operations on it.

Memory Management

By default, the SQL Server database engine always obtains as much memory as it can without causing a memory shortage on the system. It then continues to capture more memory based on factors such as the number of connections or how much work/computations need to be done. The amount of memory SQL Server uses continues to grow until either the host operating system signals that there is no more memory available or it reaches its maximum memory allocation target. Memory can be freed and reacquired as often as needed.

The DBA's View

These elements of SQL Server are important to know if you plan to talk to a DBA about your server.

Database Files

Like most other databases, SQL Server uses *datafiles* to store its data. There are always a minimum of two datafiles associated with a database: one for the database data itself and one for the logfiles. It is a generally accepted practice to locate these files on different disks or partitions to help with performance and with disaster recovery. An individual datafile can be associated with only one database.

There are three types of files in SQL Server: primary datafiles, secondary datafiles, and logfiles.

Primary datafiles

: Every database has at least one primary datafile. This file stores data and holds information about other datafiles. The default extension for primary datafiles is *.mdf*.

Secondary datafiles

: Any datafile that is not a primary datafile and not a logfile is considered a secondary datafile. These files store additional data. The default extension for these files is *.ndf*.

Logfiles

: Logfiles don't directly store any data; instead, they hold all the log information for the database. Like primary datafiles, there must be at least one logfile for each database. It is possible, and encouraged, to have

multiple logfiles and spread them across different disks. The default extension for logfiles is *.ldf*.

The physical location of the files is not very restrictive; they can be kept on any FAT or NTFS partition. NTFS is preferred because it's better integrated with later versions of Windows and has enhanced security. The files are stored in the primary file of the database as well as in the *master* database itself.

Each database file can expand as more data storage is needed. The amount of growth is specified in increments, either as a percentage or as a fixed size amount. Growth continues to occur until the maximum file size is reached. If this size is not specified, growth can occur only until the file size reaches 16 TB or until the disk is filled, whichever happens first. With more than one datafile, growth won't occur until all datafiles have been filled; growth then occurs in a round robin format starting with the first file. There is a theoretical database (not file size) limit of 1,048,516 TB.

Like databases, logfiles have various states. These states are online, offline, restoring, recovery pending, suspect, and defunct.

Filegroups

A set of related files can be grouped together in a *filegroup*. A filegroup makes it easier to manage the storage resources of the files it contains and also allows for easier administration of those files.

There are two types of filegroups in SQL Server 2005: primary and user defined.

Primary

The primary filegroup contains the primary datafile and any other datafile not specifically assigned to another filegroup. The primary filegroup houses all pages for the system tables.

User defined

User-defined filegroups are specified when the `filegroup` keyword is used in a `create database` or `alter database` command.

Keep in mind these rules for files and filegroups:

- Logfiles are never part of a filegroup; they are always managed independently from datafiles.
- Files can't belong to more than one filegroup.
- Certain database objects (tables, indexes, large objects) can be specified to use a particular filegroup.
- One filegroup will always be designated as the default.
- When creating a table or index, if no filegroup is specified, the default is used.
- Only one filegroup at a time can be the default.
- Data that resides in partitioned objects can be located in different filegroups.

Transaction Log

The *transaction log* keeps a record of every transaction (insert, update, delete, page allocation/deallocation, begin/end transaction statements) that happens in a database. The operation is recorded here before the true change is ever made to the database itself. This type of write ahead logging (WAL) guarantees that no data changes are committed to the disk before the record is

written to the logfile, which provides a redundant architecture by which problems can easily be recovered from, and transactions can be reverted or *rolled back*.

SQL Server has a logical transaction log and a physical transaction log. The *physical log* is the actual file on the disk. The *logical log* is the representation to the database engine of that log and occupies space in the physical log. The logical log can be thought of as a series of virtual logfiles of no fixed size. Space within the logical logfile is continually reused, in a circular manner, until the logfile fills. In order to prevent this amount of data from growing too large, the transaction log is periodically truncated. This truncation generally happens automatically during a backup but can also be done manually.

> **Tip ✎**
>
> Keep in mind that manual truncation breaks the log backup chain.

It is important to truncate this log to prevent it from filling up. If a transaction log fills, the database switches to read-only mode and does not allow updates. Truncation never reduces the size of the physical logfile, only the logical. To reduce the size of the physical file, special commands must be issued, which are discussed later in this section. Logfiles can also grow indefinitely if the `FILEGROWTH` setting is enabled. This allows a transaction logfile that has filled to grow by a specified growth increment. Of course, the growth of the physical logfile is limited to the amount of storage you have available.

If the transaction log does fill up, there are several options to truncate it:

- Doing a physical backup of the transaction log

- If there is free disk space, increasing the size of the logfile
- If there is not free disk space, freeing up disk space or moving the log to a drive with free space
- Adding a new logfile to a different disk
- Truncating after a system or explicit checkpoint

Monitoring logfile size with dbcc

The size of the transaction logfiles can be monitored with `dbcc`. To view the current usage, use the `dbcc sqlperf` command:

```
dbcc sqlperf (logspace)
Database Name                Log Size (MB)     Log Space Used (%)
Status
-----------------------      -------------     ------------------
-----------
master                       0.4921875         81.74603
0
tempdb                       0.4921875         63.39286
0
model                        0.4921875         73.01588
0
msdb                         0.4921875         86.50793
0
ReportServer                 0.7421875         38.68421
0
ReportServerTempDB           0.7421875         36.77632
0
Inventory                    0.9921875         45.22638
0
```

Reducing the size of the physical log

Sometimes it may be necessary to reduce the space of the physical logfile. This can be accomplished with the `dbcc shrinkfile` command:

```
dbcc SHRINKFILE (Inventory_Log,1)
DbId   FileId      CurrentSize MinimumSize UsedPages
EstimatedPages
------ ----------- ----------- ----------- ----------- ----------
----
7      2           128         128         128         128
```

Pages

SQL Server uses the concept of *pages* for data storage. SQL Server storage in datafiles is logically divided into pages, and these pages are numbered contiguously starting at 0. Disk access is managed at the page level; any read or write is done a whole page at a time.

There are 128 pages per megabyte, and each page is 8 KB. Each page starts with a 96-byte header used to hold information about the page. This information includes the page number, the allocation ID of the owning object, and the type and amount of free space.

There are page types for different storage types: data, indexes, text or image data, global, shared global allocation map, free space, index allocation map, bulk change map, and differential change map.

Rows of data are entered into the page starting after the header, in sequence. The row offset table starts at the end of the page and contains one entry for each row on that page. Each entry details how far the first byte of the row is from the beginning of the page. These entries are in reverse order of the rows on the page that they track.

Extents

Extents are important because they govern the way space is managed within the database. A group of 8 physically contiguous pages makes up an extent, and there are 16

extents per megabyte. By grouping pages in extents, the pages can be more efficiently managed.

There are two types of extents, *uniform* and *mixed*. The pages within uniform extents are owned by only one object and can be used only by that object. The pages within mixed extents can be shared by up to eight objects, meaning that each of the eight pages within the extent can have a different owner.

When creating a new table or index, space is allocated from pages in mixed extents. As the table grows past its initial size of eight pages, it then switches to uniform extents.

Partitions

Partitions segment data in a more logical and manageable manner. An additional benefit, and a substantial case for partitioning, is the potential for performance gain.

By default, tables are allocated to just one partition. Tables or indexes arranged in the default single-partition scheme are essentially the same as tables or indexes in earlier versions of SQL Server. In a table or index that spans multiple partitions, the data is partitioned horizontally. This means that groups of rows are mapped into individual partitions based on the column they are a part of. Note that partitions can be a part of more than one filegroup across the same database. Regardless of how many partitions or filegroups a table or index spans, when they are queried or updated, the object is treated as a single logical entity.

Within partitions in SQL Server 2005, data rows are arranged in one of two ways, either by clusters or heaps. *Cluster data* has a clustered index. This is a b-tree index that arranges the data rows based on the order in the clustered

index key. *Heap data* has no clustered indexes. Here, data is stored in no particular order or format.

Table and Index Specifics

Data in tables is stored in fixed-size pages of 8 KB. This data is stored in rows, and these rows generally don't span more than one page. The exception is large datatypes such as *(n)text*, *image*, *varchar(max)*, and *xml*. Large datatypes can span multiple, noncontiguous pages.

Tables span one or more partitions and, in each partition, data rows are organized in either a heap or clustered index structure. Depending on the type of data, pages of the heap or clustered index are managed in one or more allocation units. Partitions in this instance are not partitions as you would normally expect in disk storage; here they are simply a way for the user to define the way the data is organized. By default, a table or index is only associated with one partition. Keep in mind that individual partitions must exist in a single filegroup.

Snapshot Backups (2005)

New in SQL Server 2005 is the concept of snapshot backups and restore. Utilizing this technology requires software and/or hardware from third-party vendors, so we cover its benefits only briefly here because this book focuses on inexpensive backup methods.

Snapshot backups allow for very fast backups with almost no impact to the performance of the server. The major benefit of snapshot backups is that the restore process can happen equally as fast. Other benefits include creating copies that can be used for other purposes such as reporting purposes.

These types of backups are equivalent to regular backups and can be intermixed with regular backups. For instance, you can use snapshot backups in a restore process where the first full backup is from a tape. Snapshot backups are supported only with full, partial, and file backups, and partially with differential backups.

Backups

With SQL Server 2005, you can usually perform most backup operations without worrying about performance impact to users. In previous versions, greater consideration has to be given to your database architecture and backup plan so performance is not degraded. Backups can be performed at any time, but if the server is currently trying to create or delete a file, the backup is postponed until these operations are complete. Additionally, when a backup is in progress, a database cannot be created or deleted. Even though backups are not supposed to impact performance, it is always a good idea to schedule backups during the least busy time for your database.

With SQL Server, you can use many different backup techniques to minimize your downtime. This includes the creation of hot, warm, and cold standby servers. Depending on the budget and using various mirroring, log shipping, and restore processes, it is possible to create scenarios where downtime ranges from nonexistent with mirroring to possibly several hours with cold standby to someplace in between with log shipping.

Backup Devices

Backup devices can be any type of media, such as disk, tape, removable storage, or network storage (such as SAN or NAS). In previous versions of SQL Server, it was necessary to explicitly define a backup device before making a backup. In SQL Server 2005, backup devices are created on the fly. However, it is usually considered good practice to specifically define backup devices for continuity and simplification of administration.

> **Tip ✎**
>
> Unless you are using commercial backup software, SQL Server requires a tape device to be physically attached to the server on which the database instance you wish to back up is running. Backup operations to remote tape devices are not supported.

Logical and physical devices

Backup devices can be identified by either their physical or logical name. The *physical name* is the name as it would appear to the operating system, such as *d:\backups\backup.bak*. The *logical name* is simply an alias for the physical device name. The mapping of the logical device name to the physical device name is stored within the system tables. The logical name is simply an easier reference to the physical device name.

Recovery Models

A SQL Server *recovery model* helps in planning what to back up and how best to perform that backup. There are three primary recovery models available: simple, full, and bulk-logged. It is best to verify the recovery model in use for a particular database before beginning your backup and restore operations because the type of backup needed may be dictated by this setting. The recovery model affects how much data can be restored and the performance and duration of the restoration.

Simple recovery model

Simple recoveries, as the name implies, are the most basic of the recovery types in SQL Server. With this type of backup, the transaction log is automatically truncated and any inactive logs are dropped. This allows you to restore only from your last full backup because no transaction logs are available to be replayed.

Full recovery model

Full recoveries offer what simple recoveries don't because they also include the transaction logs. Replaying the transaction logs allows the database to be restored to its most recent point in time.

Bulk-logged recovery model

Bulk-logged backups are similar to full backups because they back up the existing data as well as the transaction logs. The major difference is that if this recover model is selected, any BULK transactions are not written to the transaction log. This enables you to restore all data up to the most recent point in time, except for those BULK transactions that were still present in the transaction log

The recovery model can be viewed from the GUI or by Transact-SQL.

SQL Server 2005

To find out the recovery model for SQL Server 2005, run the sp_helpdb stored procedure:

```
sp_helpdb Database_name
```

Look in the status column for the RECOVERY= section that contains the name of the current recovery model. Alternatively, you can execute the following Transact-SQL query:

```
select name, recovery_model from sys.databases;
```

This shows you a listing of all databases, followed by the number 1, 2, or 3 in the `recovery_model` column. The integer in the `recovery_model` column indicates one of the following values:

- Full recovery model
- Bulk logged recovery model
- Simple recovery model

To see the current recovery model in the Management Studio, follow these steps:

1. Open the Management Studio and connect to an instance of the SQL Server Database Engine.
2. Expand the server you wish to work with.
3. Expand the Databases group.
4. Select a user database, or expand a system database to work with.
5. Right-click on the database, and select Properties.
6. Click Options in the “Select a page” window.

SQL Server 2000

One option for finding the recovery model in SQL Server 2000 is to run the `sp_helpdb` stored procedure:

```
sp_helpdb Database_name
```

The status column for the `RECOVERY=` section will contain the name of the current recovery model. Alternatively, you can execute the following Transact-SQL query:

```
select databasepropertyex('database', 'recovery')
```

To see the current recovery model in the Enterprise Manager:

1. Open the Enterprise Manager, and connect to an instance of the SQL Server Database Engine.
2. Expand the server you wish to work with.
3. Right-click on the database you wish to view, and select Properties.
4. Select the Option tab.

The recovery model can be seen in the Model pull-down menu.

Backup Types

There are many different types of backups in SQL Server. Each has advantages and disadvantages depending on your backup strategy.

Full

A full database backup is exactly what the name suggests; it creates a full backup of the specified database. This includes all objects, system tables, data, and portions of the transaction logfiles. This type of backup allows you to completely restore the server to the state when the backup finished.

Differential

A differential backup contains all the changes since the most recent full backup. An initial differential backup with a relationship to the full backup is called the *differential base*. Over time, the differential grows, often getting close to the size of the differential base. It is therefore important to schedule both regular full backups and differentials.

Transaction log

It is important to back up the transaction log because all transactions are written to this log before ever being committed to the database. Backing up the transaction log allows you to recover the server to the most recent time since the failure. There are three different ways to back up transaction logs: a full or pure backup, a bulk log backup and a tail log backup. A *full* (or *pure*) backup contains the full transaction log for a given interval and does not include any bulk changes. Note that with this backup type, PIT recovery is not an option. A *bulk log* backup is similar to the full/pure, except it does contain transactions changed by any bulk operations. The *tail log* backup method should be used if corruption of the database is suspected. This backs up the transaction log records that have not yet been backed up by either of the previous two methods. This backup method can contain either regular or bulk logged data.

When transaction logs are restored, they are always applied in sequence, starting from the oldest and moving to the most recent. This playback is done after the most recent full or differential backup is restored and then continues through all the logs. This sequence of logs is called a *log chain*. The log chain must be intact for the entire restore process to work.

Tip ✎

Note that transaction log backups are available only in the full and bulk-logged recovery models, not in the simple model.

Copy-only backup

Copy-only backups provide a convenient way to take a copy of an existing database to use on another server. Copy-only backups do not reset the backup flag marker, so differential backups on the initial server are not affected. (The backup flag marker is similar to the archive bit in Windows.)

Partial backups

Partial and differential partial backups are new in SQL Server 2005. A *partial backup* is very similar to a full database backup except it doesn't contain all filegroups. It includes all the data in the primary filegroup as well as every read-write filegroup. It also can include user-specified files.

A *differential partial* is the same as a partial except it includes only the data in those extents that have changed since the last partial backup. As with other differential backups, the backup before the partial is called the base for the differential.

File and filegroup

An alternative to backing up your entire database is to back up files or filegroups individually. This has the advantage of possibly faster restores because if only one file fails, this file can be restored without having to do a complete database restore. Since a filegroup is simply an easy way to manage multiple files in a database, a filegroup backup is a way to more easily back up multiple files.

Backup/Restore of System Databases

System databases contain all the information about the running server's configuration as well as many other specifics such as file and filegroup location information, storage parameters, and information about system-level

configurations. Without these databases, restoring your data becomes impossible. Therefore, backup and recovery of these databases is very important.

While there are six system databases previously mentioned (*master*, *model*, *msdb*, *resource*, *tempdb*, and *distribution*), the only one required for the instance of SQL Server to start is the *master* database. If this database becomes corrupt or damaged, there are two recovery options. The first option is for when the instance can still be started. In this case, you can restore *master* from your last full database backup. The second option is if the instance is so damaged that it can't even start at all. If this is the case, a complete rebuild of the *master* database might need to be performed. After successfully rebuilding, you would restore the *master* from the last full backup.

> **Warning ☠**
>
> It goes without saying that nothing will work without the *master* database. Back up this database after every configuration change of any of your databases or after adding a new database. The *master* database backup doesn't take up much space, and it will save you a lot of grief in the end.

Viewing Information About the Backup

It is possible to view information about the history of backup operations. This information is located in the *msdb* database in the following tables: backupfile, backupfilegroup, backupmediafamily, backupmediaset, and backupset.

There are three commands to view the backup history: `restore filelistonly`, `restore headeronly`, and `restore labelonly`. The following examples use a database called *Inventory* that has a backup created in a backup set located at *E:\Backups\Inventory.bak*. To view the file list in this example, run the following command within a Transact-SQL window (Figure 19-1), which should be valid for multiple versions of SQL Server:

```
restore filelistonly from disk = 'E:\Backup\Inventory.bak'
```

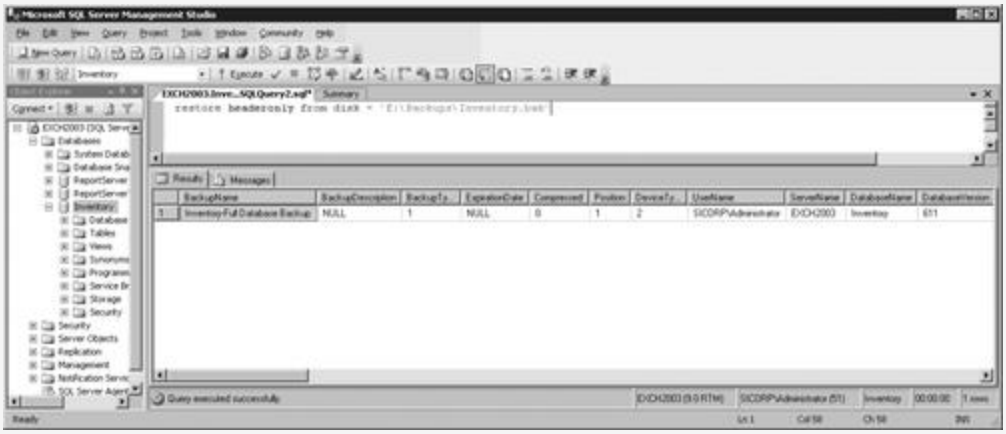


Figure 19-1. Listing the backup header information

This command lists the backup header:

```
restore headeronly from disk = 'E:\Backup\Inventory.bak'
```

Of course, the same information is also available in the SQL Server Management Studio for 2005 or Enterprise Manager for 2000:

- SQL Server 2005 SSMS
- 1. Open the SQL Server Management Studio.
  2. Expand Management, and then expand Backup Devices.
  3. Click the device for which the information is needed, and then right-click Properties.

- SQL Server 2000 Enterprise Manager
- 1. Open the SQL Server Enterprise Manager.
  2. Expand a server group, and then expand the server.
  3. Expand Management, and then click Backup.
  4. In the Details pane, right-click the named backup device, and click Properties.
  5. Click View Contents.

Verify Backups

If you have time, it is always a good idea to verify the backup once it completes. Like most other operations, there are multiple ways to accomplish this. By far the easiest is to check the “Verify backup when finished” checkbox, as in Figure 19-2 for 2005 Management Studio and Figure 19-3 for 2000 Enterprise Manager.

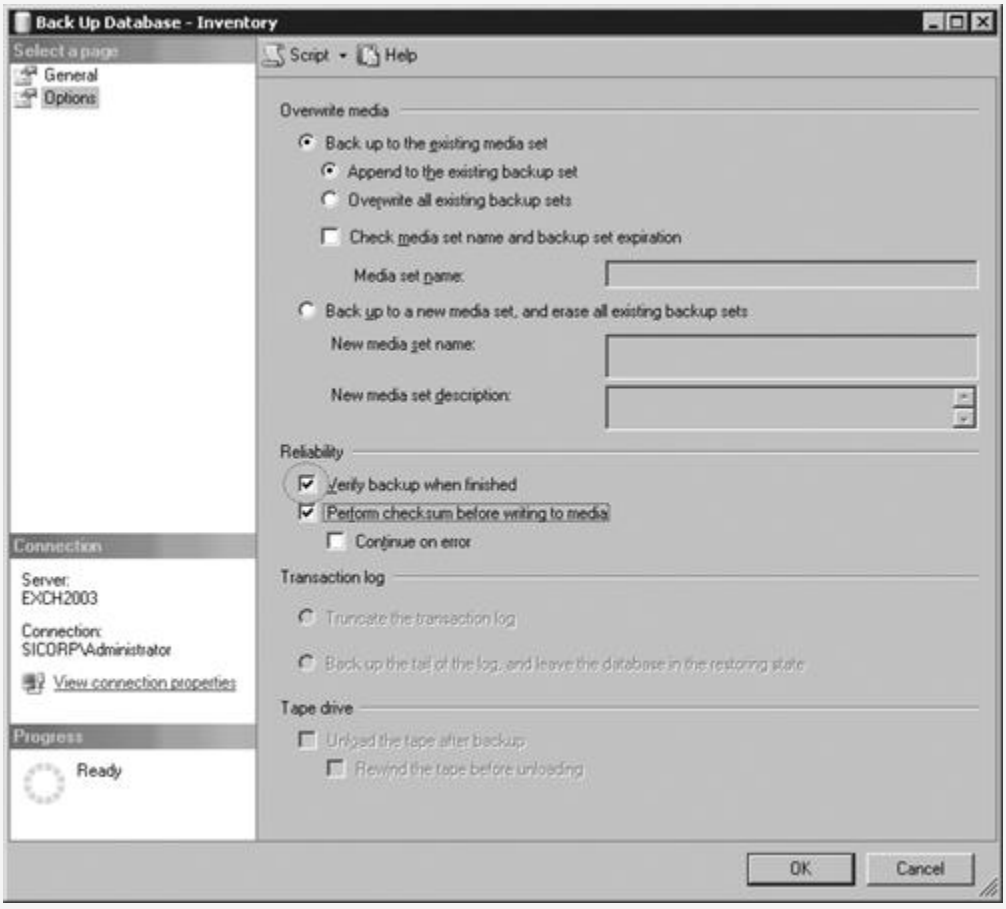


Figure 19-2. The "Verify backup" checkbox in 2005 Management Studio

Take note of the Perform checksum option below the "Verify backup" checkbox in 2005. The Verify backup option verifies only that the backup can be restored. It provides no guarantee that the actual data on the backup is valid. If the "Perform checksum" option is also selected, it provides some indication of the reliability of the data.

If you want to verify the backup using Transact-SQL, issue the following command for either 2000 or 2005:

```
restore verifyonly from disk = 'F:\Backups\Inventory.bak'
```

Backup Expiration Date

Setting the expiration date of the backup allows the backup set to be overwritten without explicitly specifying that it is

acceptable to overwrite. This date can be set using command-line backup options or from the Management Studio in 2005 or the Enterprise Manager in 2000. To specify this in the Management Studio, connect to the instance where the database is located, and then follow these steps:

1. Expand Databases, and then click on a database.
2. Right-click the database, expand Tasks, and select Backup.
3. After the Back Up Database dialog becomes visible, in the General page, specify the type of expiration—either a specific day or after some number of days.

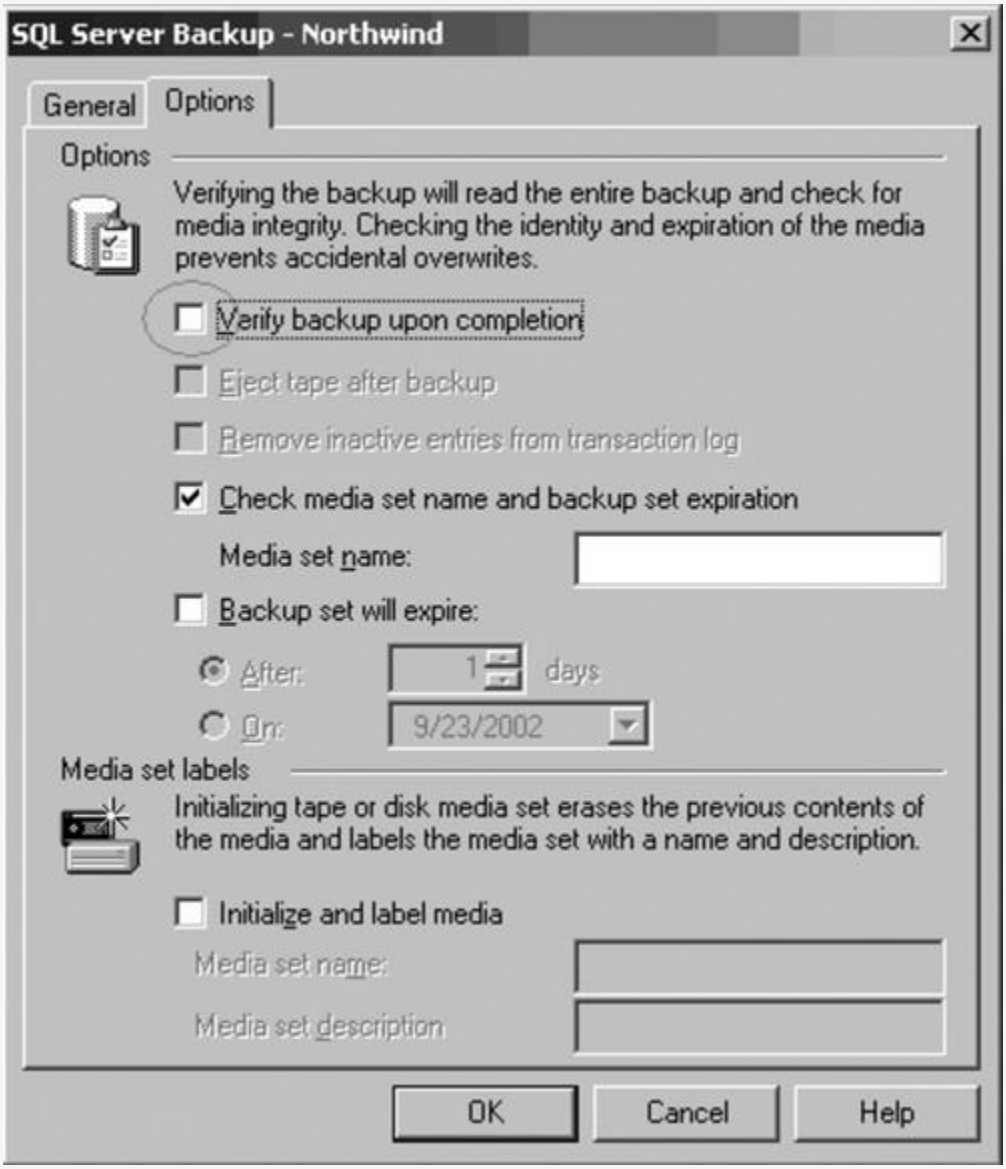


Figure 19-3. The Verify backup checkbox in 2000 Enterprise Manager

To specify this in Enterprise Manager, connect to the instance where the database is located, and then follow these steps:

1. Expand Databases, and then select the database by clicking on it.
2. Right-click the database, expand All Tasks, and select Backup Database.

3. In the Options tab, check the box labeled “Backup set will expire,” and then enter the date on which this backup will expire.

How to Back Up

In its simplest form, without taking into consideration backup strategies, recovery models, and other implementation details, the following steps perform a full backup of your database. The procedure is essentially the same between 2005 and 2000 with only a few of the options having different labels.

1. Open the SQL Server Management Studio, and connect to the instance where the database is located and expand Databases.
2. Select the database you wish to back up, right-click, expand Tasks, and select Backup.
3. The Back Up Database dialog appears, with the General page selected, as shown in Figure 19-4.

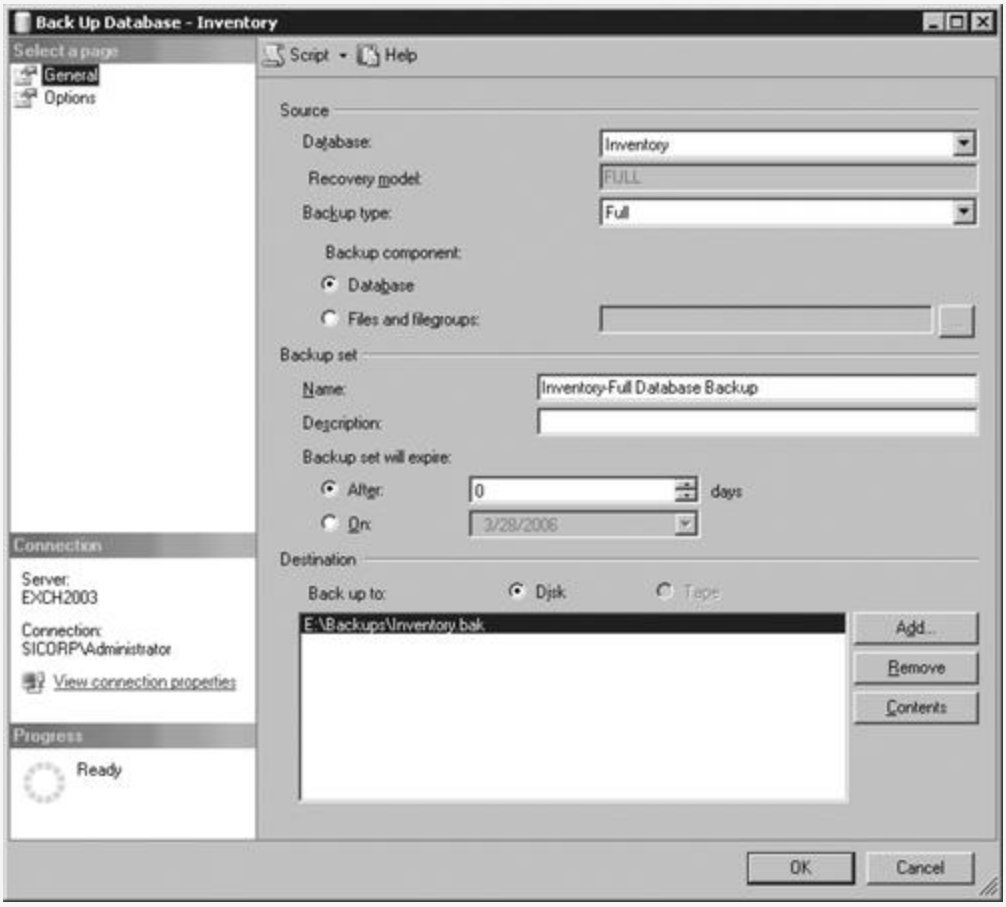


Figure 19-4. Back Up Database, General page

At first, the destination is preselected, but a diligent database administrator alters the location to be on a different disk than the actual database files. In Figure 19-4, I selected *E:\Backups\Inventory.bak* as my backup location. (The *E:* drive is a different physical drive than the *C:* drive, where the database and logfiles reside.)

Even though this is a quick backup, it would be negligent to not specify some of the options in the Options tab. At a minimum, select the checkbox to verify the backup, as previously shown in Figure 19-3.

Of course, this performs only the most basic of backups. Ideally, careful consideration would be given to planning your backups. This includes full or differential backups,

locations of these backups, how much storage is necessary, and other details.

Command-line backup with Transact-SQL

The same disk-based backup can be accomplished using Transact-SQL and the `backup database` command for 2000 or 2005. The following command does a basic backup without any additional options:

```
backup database inventory to disk = 'E:\Backups\cmd_Inventory.bak'
with name = 'Command Line Inventory Backup'go
```

Transaction Log Backups

It is also important to ensure that the transaction logs are backed up on a regular basis. This can be done only with a full or bulk logged backup type. In order to perform a transaction log backup, follow these steps:

1. Open the Management Studio, connect to the instance where the database is located, and expand Databases.
2. Select the database you wish to back up, right-click, expand Tasks, and select Backup.
3. Verify that “Recovery model” is set to Full or Bulk Logged.
4. In the Backup Type drop-down list, select “Transaction log.”

These steps are shown in Figure 19-5. They are almost the same for 2000 with Enterprise Manager, but the Backup type is a set of radio buttons instead of a drop-down list.

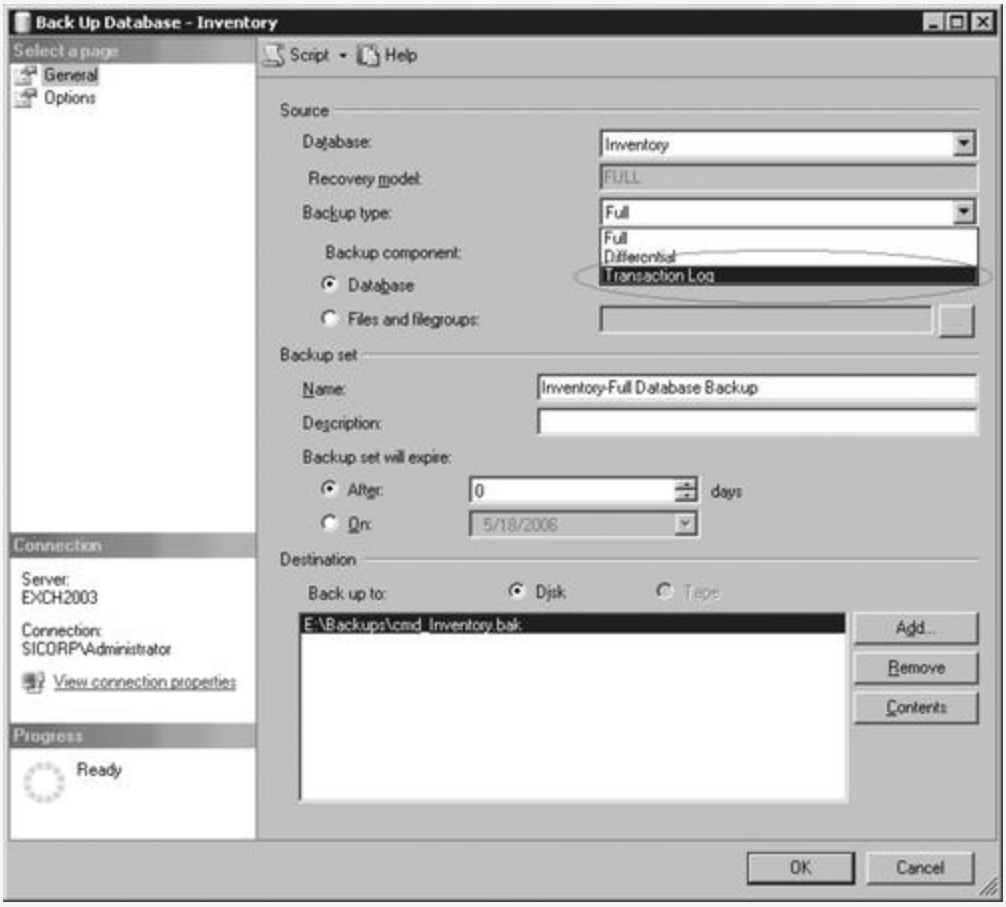


Figure 19-5. Transaction log backup

Go through the rest of the options on the page and customize as needed. Afterwards, select the Options page, and make changes as necessary to the Transaction log section, shown in Figure 19-6. If this is a normal backup, simply leave the default “Truncate the transaction log.” Select the “Back up of the tail log” option only if you are doing a tail log backup of the database after a failure, are in preparation for a restore operation, or are in preparation to fail over to a secondary database.

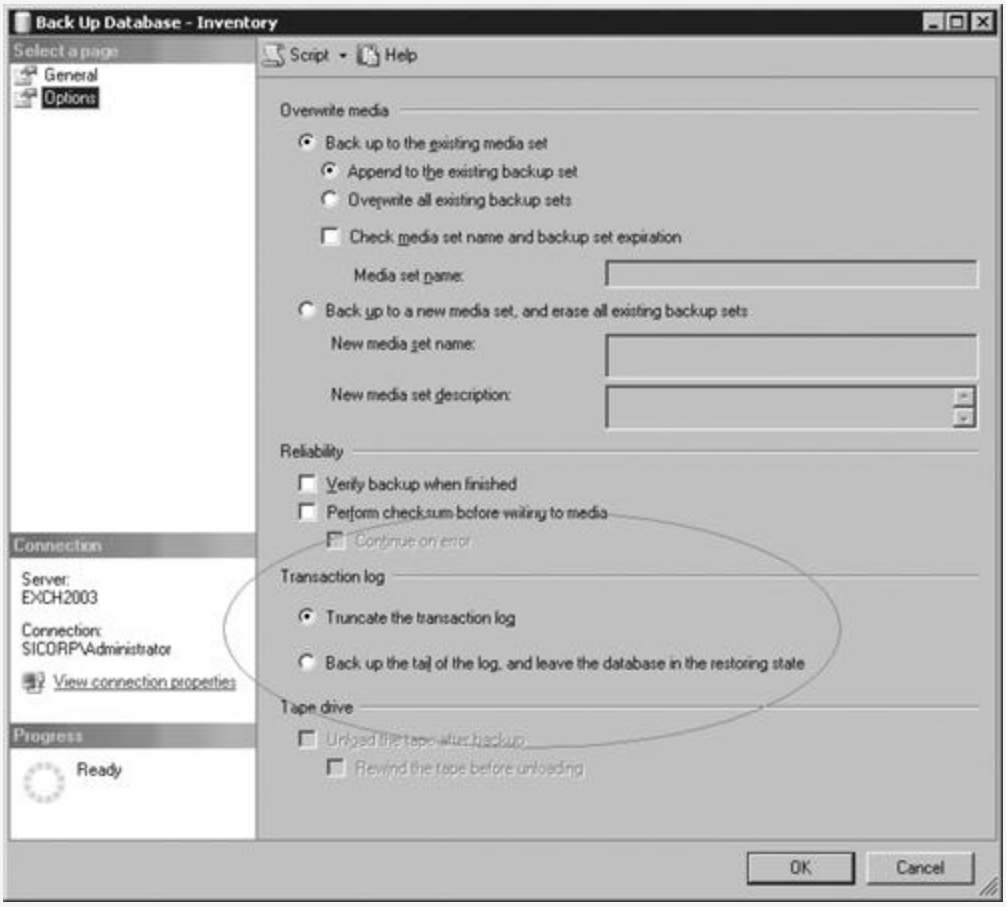


Figure 19-6. Transaction log options

As with most other operations, the backup of transaction logs can also be accomplished using Transact-SQL as in the following example. This example backs up the transaction log for the *Inventory* database to the previously created backup device *Inventory_dev*; this procedure is valid for 2005 and 2000:

```
backup log Inventory to Inventory_dev
```

Log truncation

Logs could theoretically grow until they fill the entire disk. To prevent this, logs are periodically truncated. Depending on the recovery model, logs are truncated at different times and in different ways. In the simple recovery model, in

which log backups are not supported, log truncation is automatic, usually happening after a checkpoint. In the full and bulk-logged recovery models, the log is truncated when it is backed up.

Master Database Backups

It is extremely important to back up the *master* database on a regular basis. This database holds all the configuration information for the running system as well as all the configuration information for all databases and other information such as logon accounts. Without this database, the rest of the system is useless!

> **Tip ✎**
>
> Only full database backups are supported for the *master* database.

Scheduling a Backup

To save time running manual backups, SQL Server allows jobs to be created that can automatically back up your database at a specified time. An automated job consists of two pieces: the backup job itself and a backup schedule. To create a new job, follow these steps for 2005 Management Studio:

1. Open the SQL Server Management Studio, connect to the instance where the database is located, and expand Databases.
2. Expand SQL Server Agent.
3. Right-click on Jobs, and select “Create new job.”

4. Type in a name for the job, and make sure Enabled is selected.

This creates the backup job, but it is also necessary to create a schedule. That can be done from the initial job creation screen by selecting the Schedules page, or if you've closed it, by right-clicking on the backup job, and selecting Properties.

Once you've clicked on the Schedules page, select New to open the window shown in Figure 19-7. From here, enter a Name, and set the type of backup frequency.

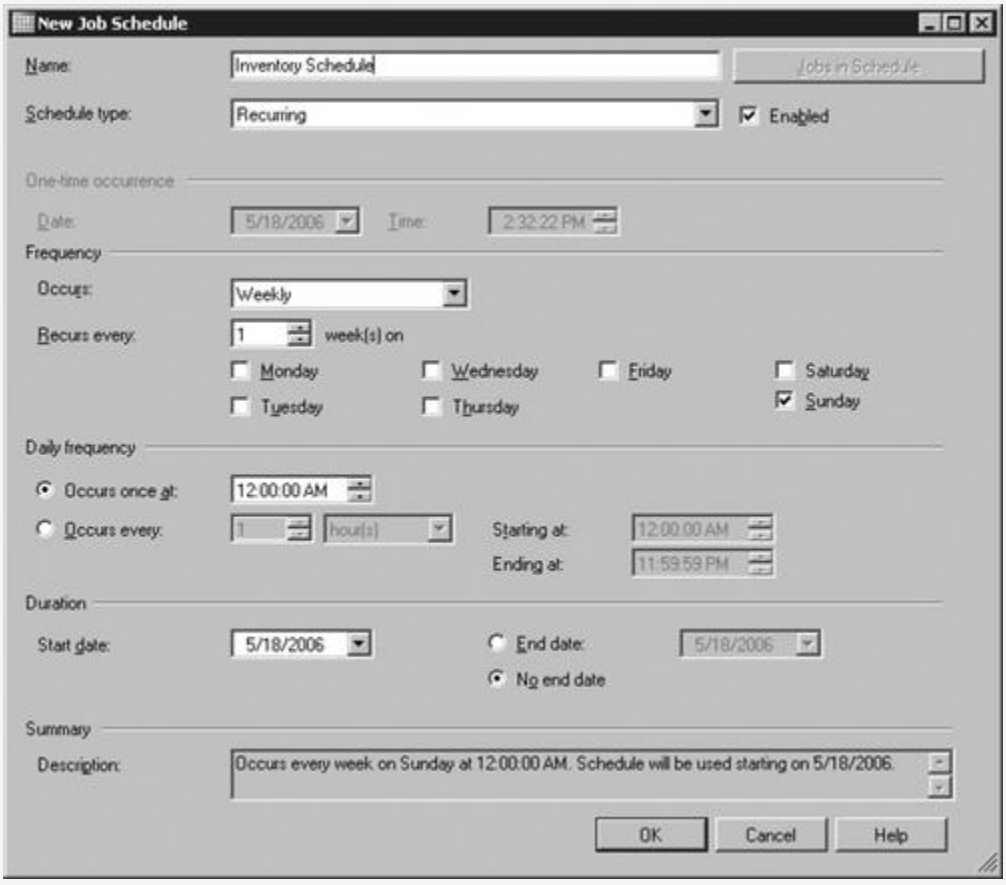


Figure 19-7. Scheduling a backup job

In 2000's Enterprise Manager, the easiest way to schedule a backup is to use the Maintenance Plan wizard:

1. Open the SQL Server Enterprise Manager, connect to the instance where the database is located, and expand Databases.
2. Right-click on the database, select All Tasks, and then select Maintenance Plan.

This wizard walks you through the steps necessary for scheduling a backup.

Logical (Table-Level) Backups

One alternative to SQL Server backups is to dump the data into text format to be later inserted into some other database, a spreadsheet, or even back into SQL Server. This is accomplished with the bulk copy utility, `bcp`. The major benefit of this tool is that it doesn't require any knowledge of Transact-SQL, though Transact-SQL can be used to refine the data output. This utility is available for all versions of SQL Server.

The following example dumps all the data in the Items table in the *Inventory* database to a file called *Items.dat*.

```
bcp inventory..Items out Items.dat –T –c
```

A similar operation can be performed to insert the data into a table. In this example, simply insert the data back into the table it originated from.

```
bcp inventory..Items in Items.dat –T -c
```

Restore and Recovery

In SQL Server terminology, a database restore and a recovery represent two different operations. A *restore* is the act of copying data from a backup, applying the logged (committed) transactions to roll the database forward to the specified recovery point desired, and rolling back any uncommitted transactions that were found in the transaction log. *Recovery* happens when the database starts. It rolls back any uncommitted transactions so that the database starts in a consistent state. If the recovery process is unable to return the database to a consistent state, a restore may be required.

Just as there were a great many choices in deciding what and how to back up, equal consideration must be given as to how you will perform your restore. Your restore will usually be dictated by the last backup you did. For instance, the best-case scenario is that you just completed a full backup and now need to do a full recovery. This simply involves the last full backup without the need to apply differential or transaction log backups. It is easy to see how the number of choices can quickly increase. You could do a full, a full with a differential, a full with transaction logs, or a full with a differential and transaction logs. Which type of restore you need to do involves careful consideration of how much data you need to restore and what current backups are available.

Components of a Restore

Various components make up the restore. These include the rollforward set, restore sequences, and backup recovery.

Rollforward set

The rollforward set consists of all the data that is to be restored, regardless of whether the data is from a full, partial, or other type of backup.

Restore sequence

A restore sequence is defined as the group of backups that is restored to bring the database online and to a consistent state. These backups are restored, in the proper sequence, to accomplish this.

Backup recovery

The actual restore process has three phases, including data copy, redo, and undo. The *data copy* phase copies all the data, logs and index pages from the backup to the database file. In the *redo* phase, also called the *rollforward* phase, the logged transactions are applied to roll forward the data to the specified recovery point. After this phase, the database is generally in an inconsistent state, needing the undo phase in order to complete. In the *undo* phase, which is where the backup recovery process starts, any uncommitted transactions are rolled back, bringing the database to a consistent, usable state. Note that the undo phase isn't always necessary if the rollforward operation returns the database to a usable state. During this process, different safety checks are performed by the database engine. These checks prevent overwriting existing databases or existing datafiles.

Recovery Roadmap

The process of restoring or recovering your server can be difficult and confusing. Trying to determine where to begin is probably the biggest hurdle to overcome. This roadmap walks you through the basic steps involved in trying to get

your server running again. The restore and recovery process makes no assumptions about what the failure was or the current state of the database.

Before jumping right in and trying to recover the database from the last backup you have, it is always a good idea to ensure that the database is really in a corrupt or unusable state. There are several ways to confirm this. After these first steps have been completed, move on to the actual restore and recovery.

Step 1: Check for obvious hardware errors or server problems

While this category is too broad to cover in depth, it is always best to ensure that the server is in proper working order. This means all hardware is working and, if you need to reinstall the OS, that all patches are applied.

Step 2: Can you connect to the instance using a GUI or T-SQL?

Your first step in locating the problem should be the basic sanity test of connectivity. This will help you narrow down any major problems by allowing you to do the additional tests in the following steps. If you can't connect, see if the instance is running, and if it isn't, start it. If there are no obvious reasons why the instance won't start, begin your troubleshooting with the event log and any logfiles. After you can connect to the instance, proceed to Step 3.

Step 3: Can you connect to the master database?

After you're sure the instance is running, the next step is to verify that you can connect to the *master* database. Remember that no other databases can run without the

master! If you can't connect to the master, follow the steps outlined in the *master* database repair section. After you can connect to the master, proceed to Step 4.

Step 4: Can you connect to a specific, nonsystem database?

Now we are getting to the meat of the problem. If you're at this step, you know your hardware and OS are good, your instance is running, and the *master* database is operating normally. If you can connect to a nonsystem database, you're done! However, it probably won't be this easy. The first step is to check to see the status of the database. This procedure is accomplished with the following command:

```
select databasepropertyex('Inventory', 'Status');
```

This returns a value that represents the current status of the database. In this example, the possible values are online, offline, restoring, recovering, suspect, and emergency. These values provide a quick glimpse of the status of the database. The level of effort involved in getting your database running will be directly related to this result. For instance, if the status is offline, it might be as simple of starting the database. However, if the level is suspect or emergency, it could take a lot more effort to determine the cause of the problem. After you've determined the status, continue to Step 5 to start the troubleshooting process.

Step 5: Initial checks

The first steps in locating the problem are to look for obvious errors in the event log and any other error logs. Browse the event log for specific errors that would lead you to the root of the problem. Also be sure to check the logs

generated by the agent process, especially the error logs and the alerts.

Another option here is to run `dbcc` with the `checkdb` command or one of the other `check` commands:

```
dbcc checkdb ('Inventory');
```

If the `dbcc` utility reports any errors, start with this as a base to getting your database running. There are many different options for checking objects with `dbcc`. Entire databases or even tables can be checked, as was just demonstrated, with `checktable`. Other objects can be checked with `checkalloc` and `checkcatalog`, which check the consistency of disk space allocation and catalog consistency, respectively. These commands should hopefully lead you to one of the other steps that will allow you to repair or restore your database. After this step, proceed to Step 6 if your database is still not working.

Step 6: Are any datafiles missing?

This is a fairly straightforward test. Try to start the database using your preferred method. If you see an error such as the one in Figure 19-8, which can be seen when you're trying to start the database from the GUI, you'll know that there is a missing file.

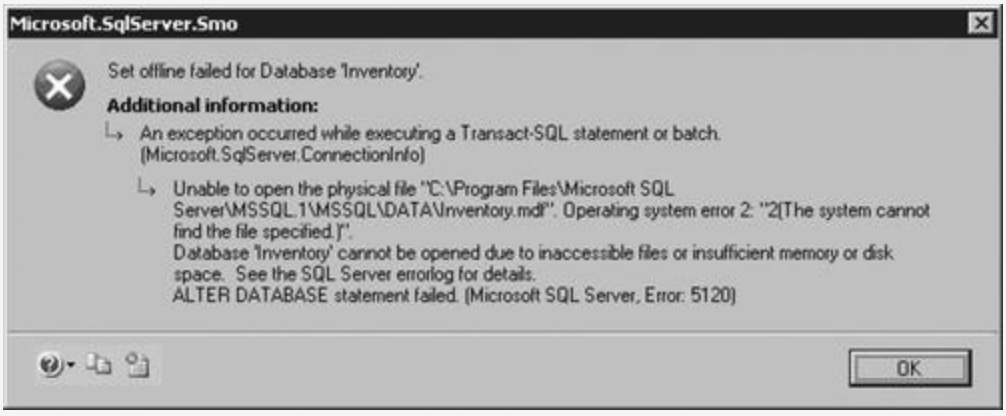


Figure 19-8. Missing datafile

From T-SQL, you would receive a similar error after issuing the following command:

```
alter database Inventory set online;
Msg 5120, Level 16, State 101, Line 1
Unable to open the physical file "C:\Program Files\Microsoft
SQL Server\MSSQL.1\MSSQL\DATA\Inventory.mdf".
Operating system error 2: "2(The system cannot find the file
specified.)".
Msg 945, Level 14, State 2, Line 1
Database 'Inventory' cannot be opened due to inaccessible files
or insufficient memory or disk space.  See the SQL Server
errorlog for details.
Msg 5069, Level 16, State 1, Line 1
ALTER DATABASE statement failed.
```

You can try to restore this datafile to bring the database back online. However, keep in mind that this may or may not bring your database to the most recent point. You may still need to perform additional restores of incremental or transaction logs. If restoring the datafile does not render your database operable, proceed to Step 7.

Step 7: Is the transaction log full?

If you have a database with a high level of activity, the transaction log might be full. It is also possible that the log is full because either your disk is full or you don’t have auto-growth enabled on that logfile. Make sure before you begin any of these fixes that you make a backup. This will ensure that, no matter how wrong the restore goes, you’ll always be able to revert back to the last known state. Yes, this state is the failed state, but at least you’ll be able to begin again.

If the disk has filled up, you can try to gain a temporary reprieve by truncating the transaction log, doing a backup, or freeing space on the drive by removing unnecessary space abusers. However, this will likely buy you only a

short amount of time. The long-term solution is to move the transaction log to a new disk.

If the disk hasn't filled up, but the transaction log has filled up, you can try to enable auto-growth if it is not enabled. If it is enabled, it may be necessary to add another transaction logfile. Steps to see if the transaction log is full are outlined in the section "Transaction Log," earlier in this chapter. If this isn't the reason for your database failure, continue your quest at Step 8.

Step 8: Is it possible to repair the DB?

One possible repair alternative is to use `dbcc` with the `repair` option. While I mention it here, I won't go into detail because this method often results in data loss (though not always). Consult the `dbcc` reference page for specifics on this method. Proceed to Step 9.

Step 9: Before you begin the restore process

The most important thing before you begin the restore process it to determine what recovery model is in use. This setting directly impacts the type of restore you'll do. Consult the section "Recovery Models" earlier in this chapter for directions on how to determine the current recovery model.

> **Warning ☠**
>
> No matter which recovery model is in place, always take a backup of your existing system. This ensures that if something goes wrong with the restore, you can at least get back to the point where you began.

If the recovery model is simple, proceed to Step 10.

If the recovery model is full or bulk logged, proceed to Step 11.

Step 10: Restore under the simple recovery model

A restore under the simple recovery model is quite, how shall we say, *simple*—hence the name. Since it doesn't deal with transaction logs, it is straightforward. Restore the last full backup and any differential backups, and you should be finished!

Step 11: Restore under the full or bulk logged recovery model

Under the full or bulk logged recovery model, restores can be slightly more complex than in the simple recovery model due to the fact that you may also need to apply transaction logs. Here are the steps:

1. Restore the last full backup.
2. If you have differential database backups, restore them too.
3. Restore and replay the transaction logs.

> **Warning ☠**
>
> Back up both the database and the transaction logs before your restore!

Database Restore

This section covers how to do a restore of the full backup created in the section "Backup" earlier in this chapter. This

procedure outlines the steps necessary for a restore in 2005 Management Studio:

1. Open the SQL Server Management Studio, connect to the instance where the database is located, and expand Databases.
2. Select the database to restore from, right-click, expand Tasks, select Restore, and then select Database.
3. On the General page, select the database you wish to restore to. If the database does not exist, type in a name for the new database.
4. Select the point in time you wish to use. Unless you need a specific previous date, Most Recent Possible (the default) should suffice.
5. Specify the source location of the restore, specifying either a database or a device.
6. Select the backup sets to restore, and click OK.
7. Specify options on the Options page:
8. a. Specify whether the restore should overwrite the existing database.

   b. Specify the recovery state desired after the restore. The recovery state you select depends on any operations that need to be performed on the restored database after the restore is complete. If your backup contains all the information you need to do a full restore, select Leave The Database Ready To Use. If, however, you need to do more operations on the database before bringing it online for others to use, select one of the other two options: Leave Database Non-Operational or Leave Database In Read-Only Mode. The first, Leave Database Non-Operational, is akin to a manual

restore. It restores any backups selected—full, differential, or transaction logs—and then leaves itself nonoperational so that more work can be done. This can include applying more transactions or other related restores. Leave Database In Read-Only Mode is similar to nonoperational mode except that after the restore is complete, the database is brought up in read-only mode to allow additional transaction logs to be applied. In this mode, you can run checks on the database and verify the data

The Restore Database window is shown in Figure 19-9. The procedure is essentially the same between 2005 and 2000 with only a few of the options having different labels.

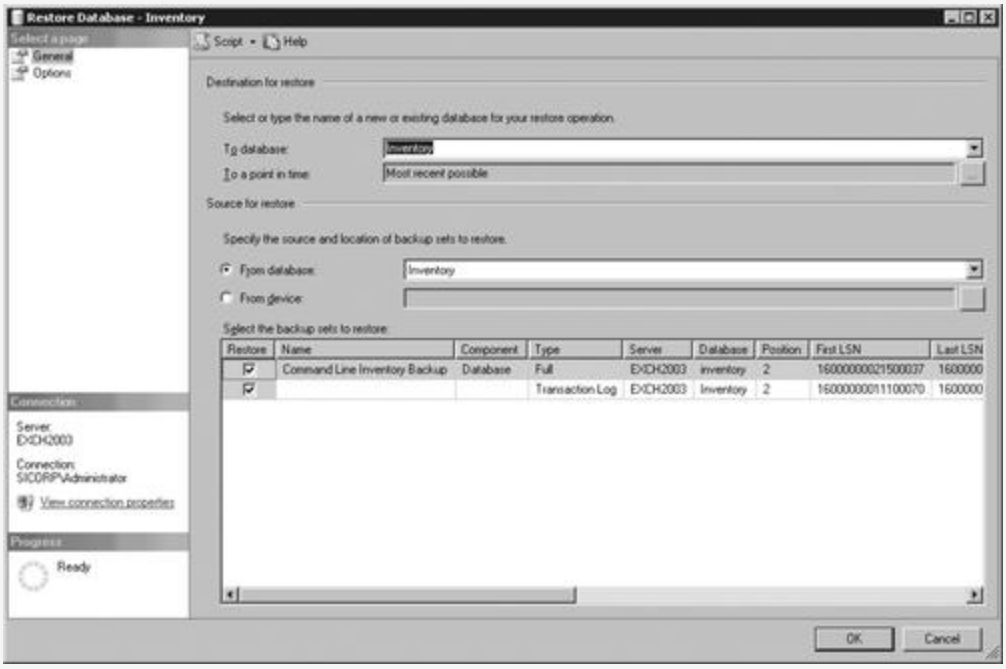


Figure 19-9. Restore Database window

This example was an illustration of the most basic of full restores. Depending on your backup plan, it may, of course, be necessary to restore differential backups and transaction logs.

Command-line restore with Transact-SQL

As with backup, you can do a restore from Transact-SQL using the `restore database` command. The following Transact-SQL command restores the *Inventory* database:

```
restore database Inventory from disk = 'E:\Backups\cmd_Inventory.bak'
```

> **Tip ✎**
>
> It is also possible to restore individual files and filegroups that you have backed up. If you have designated objects to span more than one filegroup, you must restore every file that that object touches.

Master Database Restore

Because the *master* database contains the configuration information for all the other databases, as well as the configuration information for the server, it is the most important database to back up and be able to restore. Unfortunately, it is not always easy to determine whether the *master* database is the problem. After all, if you can't connect to any instance, how can you check to see whether the database is OK? There are two ways to recover from this type of problem.

If SQL Server can start, you can connect and restore the *master* database. This restore process is just like the restore for any other database. Keep in mind that this can only be a full restore because that is the only method supported by the master database. Also note that since differential or transaction logs are not able to be restored, the master database may not be in the exact state as when the

database failed. To remedy this, manually apply any changes made since the last full backup.

If SQL Server can't start, you can rerun the SQL Server Setup program and have it rebuild the *master* database. Afterward, you can restore from your latest backup. As in the previous example, you may need to do some manual updates to get the database to the state before the failure. The Rebuild Master option in the Setup program also rebuilds the *msdb* and *model* databases.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 20. Exchange

Exchange Server is used to send and receive messages and other content using interconnected computer networks. In addition, it boasts a robust collection of collaboration tools to enable groups to work more effectively together. All of this functionality is built on a specialized database that allows for advanced functionality and configurability. In this chapter, we'll examine the specifics of Exchange Server (from now on simply referred to as "Exchange") and cover proper backup, restore, and recovery procedures.

> **Tip ✎**
>
> This chapter was contributed by Scott Harris. Scott is a brand-new father who's wondering how old his son Zach needs to be before he starts teaching him how to write PHP.

Exchange Architecture

To understand Exchange backup and recovery, you need a basic understanding of its architecture. We'll cover the design of the backend engine that makes Exchange what it is and then continue on to examine the details of how it works.

Database Structure

Exchange is essentially a specialized database for processing messages and similar content. In understanding how Exchange structures the underlying files and databases, it is useful to have an overview of the

architecture. The principal database technology used by Exchange is called the Extensible Storage Engine (ESE) and is a specialized database based on Microsoft's Joint Engine Technology (JET). The current version of this JET database in use for Exchange Server 2000 and 2003 is ESE98. (Active Directory, which is tied closely with Exchange, uses ESENT.)

> **Tip ✎**
>
> While there are differences between different versions of Exchange, most of the information presented in this chapter is valid for 2003 and 2000 at a minimum. All screenshots reflect configuration parameters taken from Exchange Server 2003. Any major differences are noted.
>
> At this writing, Microsoft is still maintaining that it plans at some point to use SQL Server technology as the underlying database for Exchange. When that will happen is left as an exercise for the reader.

The ESE database stores its information in files. Two files in particular, *.edb* and *.stm* files, are tied together to form a *store*. Stores can be grouped together to form a *storage group*. A storage group maintains a common set of transaction logs for all stores within the storage group. This is illustrated in Figure 20-1.

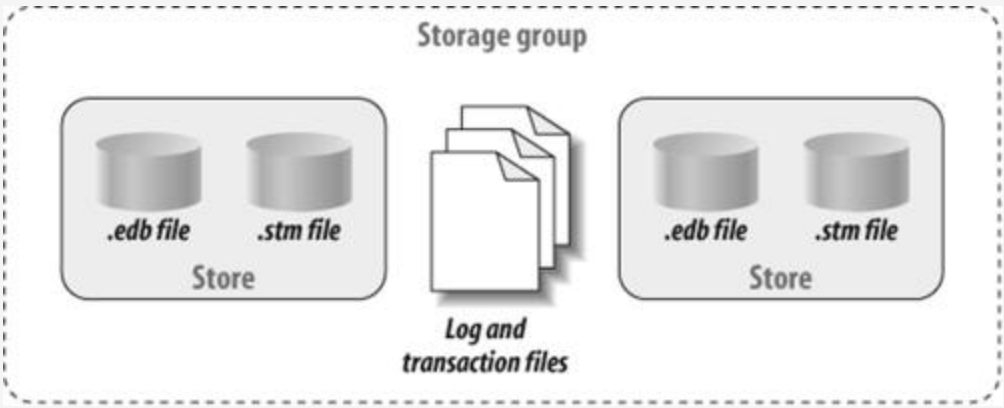


Figure 20-1. Common transaction logs for all stores within the storage group

Now that we have a basic understanding of how things are arranged inside an Exchange Server, we can look at the details of each component.

Extensible Storage Engine

Beneath Exchange is a transactional database similar to other commercial databases. The ESE provides all the usual Data Manipulation Language features you would expect in a high-powered database: inserts, updates, selects, and deletes. These DML statements are all wrapped into atomic units called transactions. *Transactions* are the basis for ensuring that database integrity remains intact by providing a means for rolling back a group of statements if a problem arises or committing the transaction otherwise.

To demonstrate the transactional nature of Exchange, let’s walk through the steps for a simple message move between the inbox and another folder located on the server (not a local *.pst* file). Because the ESE works with transactions in snapshot isolation mode, every transaction is guaranteed a unique view of the database during the operation. To begin the transaction, the commands to erase the message from

the inbox are written to an in-memory structure called the *version store*. If this step is successful, the commands to write the same message to the second folder are written in the version store. After these steps have completed without error, the transaction log is played back, actually causing the data move to physically take place. If no errors occur at this point, the transaction is committed. If there are errors, the transaction is rolled back, and no changes are made.

Stores

Now that we know how Exchange accesses data, we need to know where it stores this data. Exchange maintains its databases in objects called *stores*. There are two types of stores, mailbox stores and public folder stores. Each store consists of two files, rich-text database (*.edb*) files and native content database (*.stm*) files. *.edb* files are specialized to store Messaging Application Programming Interface (MAPI) messages, checksums for the *.edb* and *.stm* files, and tables used by the *store.exe* process. *.stm* files store native Internet-formatted content such as Multipurpose Internet Mail Extensions (MIME) content. While *.edb* files handle most of the data, the primary purpose of the *.stm* database file is to support non-MAPI clients. Figure 20-2 shows where the store filenames are located in a default Exchange install. You can see that this is the default mailbox store, and its associated *.edb* and *.stm* files are created under *c:\Program Files\Exchsrvr\mdbdata* in this example. These locations are the same for 2000 and 2003.

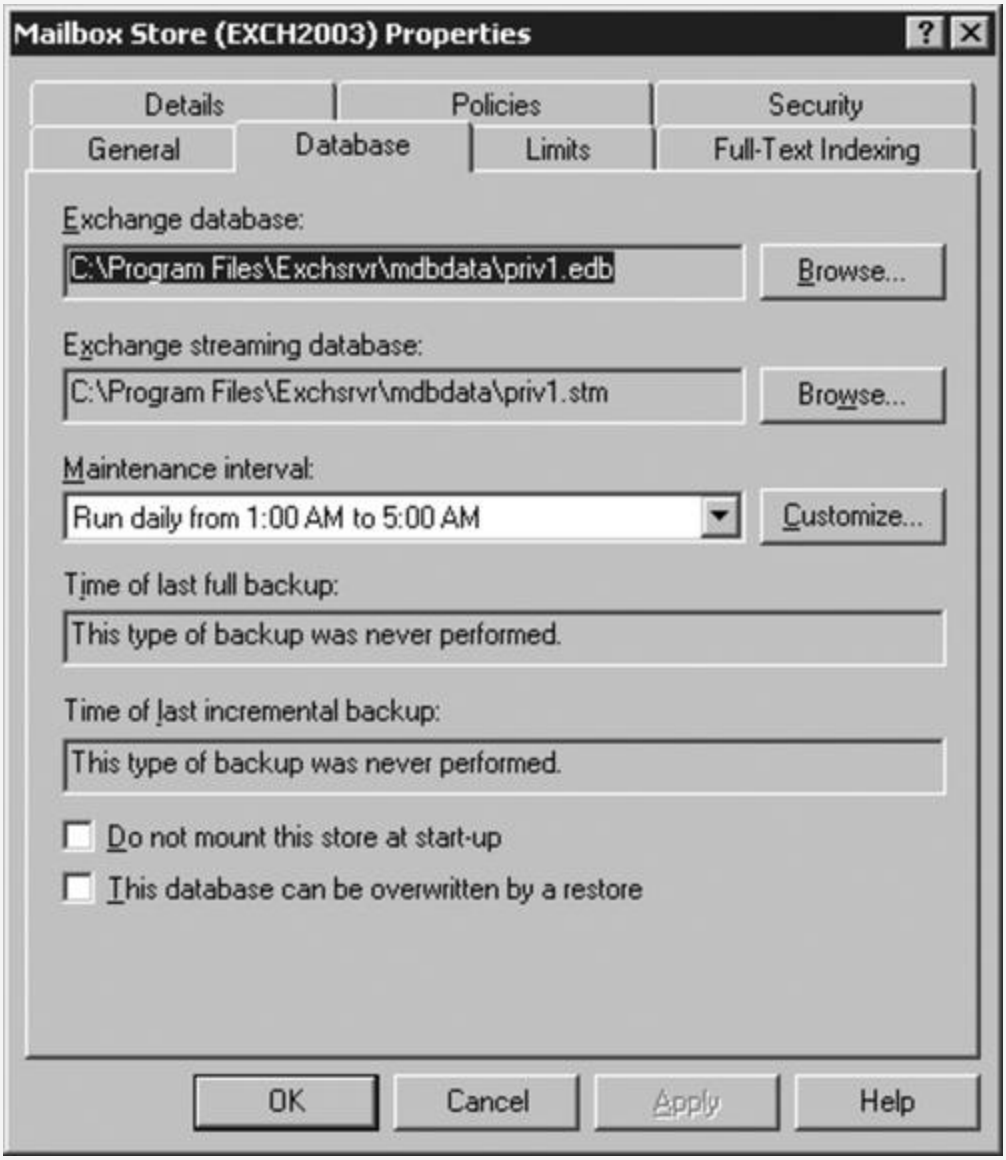


Figure 20-2. Location store filenames

All Exchange installations have a mailbox store and a public folder store. The mailbox store contains user private data whereas a public folder store contains public or shared information. The database files in the default mailbox store are named *priv1.edb* and *priv1.stm;* the database files for the public folder store are named *pub1.edb* and *pub1.stm*. With Exchange Server 2003 Standard Edition (pre-Service Pack 2) and Exchange 2000, these databases can hold 16 GB. Standard Edition with SP 2 is increased to 75

GB per store. In Exchange Server 2003 and 2000 Enterprise Edition, the databases are limited to 16 TB. Figure 20-3 shows the default storage groups and stores. For reference, the name of the server in these examples is *EXCH2003*.

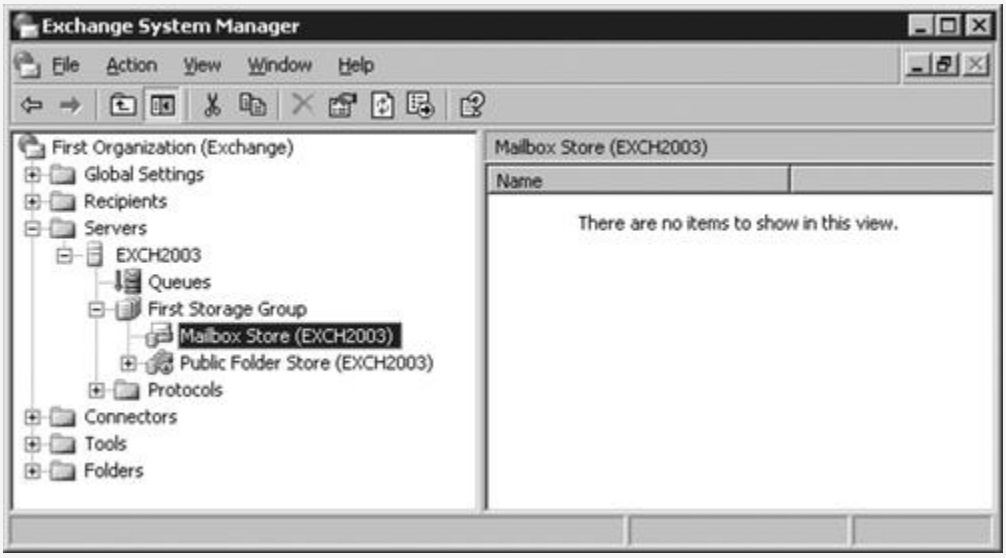


Figure 20-3. Default storage groups and stores

Storage Groups

Storage groups are used to house one or more groups of stores. Exchange Server 2000 and 2003 can have up to four separate storage groups, and each can contain up to five stores. Storage groups are used to reduce the management of the number of logfiles and thereby also the amount of transactions on those files that would be necessary if each store had individual logfiles. Additionally, by grouping stores together, they not only share the same set of logfiles but they can be managed in the same way. This includes sharing backup settings or backup schedules. Storage groups consist of many elements, including transaction logfiles, checkpoint files, reserve logfiles, and temporary files.

Transaction Logfiles

The transaction logfile maintains a history of every operation occurring in a specific storage group. Transactions are stored sequentially in the logfiles and concurrently in memory. Transactions are simply appended to the log for simplicity and speed, and then replayed into the database at a later time. This has the added benefit of not only being fast, but providing a recovery mechanism if the server should fail. If you lose the transaction in memory, you have a record stored in the logfile.

Exchange transaction logfiles are always 5 MB. Having fixed-sized logfiles offers a performance benefit over growing logfiles because the overhead associated with having to grow or shrink a file is eliminated.

When Exchange starts, it creates one 5 MB transaction logfile named *E<xx>.log* where *<xx>* is a sequential number

starting at 00 referred to as the base name or log prefix. For instance, the default storage group that is created with every Exchange Server installation is called, cleverly, First Storage Group and its logfile is named *E00. log*. If you were to create a second storage group, its logfile would be named *E01.log*, and so on. Figure 20-4 shows the default storage location and log prefix in Exchange Server 2000 and 2003.

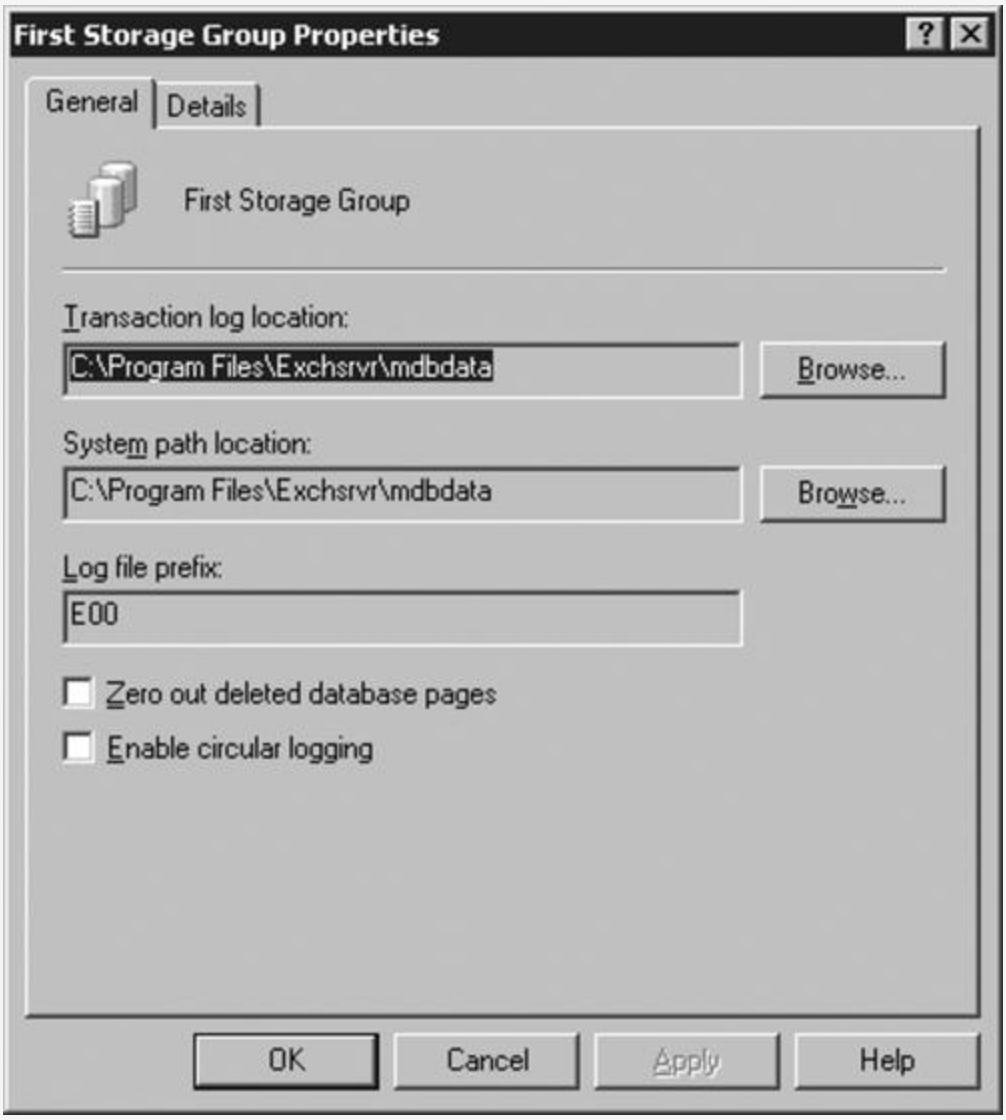


Figure 20-4. Default storage location and log prefix

Eventually, this file fills and ESE renames it to *E<xxyyyyy>.log* and creates a new file with the same name as the original

E<xx>.log. Logfile numbering is hexadecimal starting at 00001, so in staying with the naming used in previous examples, the first logfile created would be *E0000001.log*. This process is repeated each time a logfile fills up until one of two events occur: either you reach the maximum number of logfiles allowed, the server logs an error in the event log, and the store is gracefully dismounted; or you make a backup using one of the backup methods that purge unnecessary transaction logfiles.

Checkpoint Files

Checkpoint files are important because they keep track of how far Exchange has progressed through the process of writing the transaction logfiles to the database. Without this record, there would be no way to know where to start replaying transactions during a restore or recovery. This would mean having to replay every single transaction since your last backup.

Checkpoint files are named in a similar manner to the primary transaction logfile; they inherit the base name/log prefix and are named *E<xx>.chk*. The name of the checkpoint file for the First Storage Group would be *E00.chk*. There is never more than one checkpoint file per storage group, and it is always 8 K.

The checkpoint file maintains a pointer to the oldest logfile in the storage group that has all transactions successfully committed to the database. When all transactions in a logfile are written successfully to the database, the checkpoint advances to the next logfile in the series that contains the next unwritten entry.

If there is a server or database failure, ESE reads the checkpoint file on startup to find the correct transaction logfile to recover any lost transactions. ESE recovers the

lost transactions by writing to the database all transactions that are newer than the checkpoint file. Any uncommitted transactions are discarded.

As mentioned previously, the *E<xx>.log* file is not required to replay transactions. ESE can determine which transactions are already written by examining the transaction logfiles, in sequence, one at a time. Using the checkpoint saves ESE from starting at the first logfile and possibly having to progress through hundreds or thousands of files to find the proper point at which to start replaying transactions.

Reserve Logfiles

For each storage group, ESE reserves two logfiles named *res1.log* and *res2.log*. These files are used as placeholders in the event that the server runs out of disk space and can't create any more transaction logfiles. These logfiles have the same size and function as normal logfiles; they are simply held in reserve. They are called into use only if the Exchange Server has less than 5 MB of disk space available and can't create a new logfile. At this point, ESE would then use one of these reserved logfiles instead of creating a new logfile. Operations currently in memory are recorded to the reserved logfile starting with *res2.log*, and then the databases in the storage group are dismounted. Of course at this point, there is no access for users, and you'll be able to track the problem from the errors in the Event Log.

General Logfile Info

Logfiles can have the same name in different storage groups. To track which storage group each logfile belongs to, ESE adds a header to each file series with a unique signature to differentiate between different series. These

signatures do not change after the *E<xx>000001.log* file is created, to maintain consistency. In general, the rotation of logfiles works like this: the current *E<xx>.log* reaches capacity; at this point, the *E<xx>temp.log* file is started. The original logfile is renamed to its hexadecimal number and then finally the *E<xx>temp.log* file becomes *E<xx>.log*. This process is repeated on an as-needed basis.

Circular Logging

Circular logging allows transaction logfiles to be overwritten by new logfiles after the transactions contained within are committed to the database. This is a great feature for frontend-pass through type servers, where no information is actually stored, because it maximizes disk space usage. The main disadvantage to using circular logging is that if the database fails, recovery can't replay transaction logs, so your restore is limited to only the information contained in the most recent full backup. Also, with this type of logging, you are limited to full backups; differential or incremental aren't supported.

Circular logging works by not maintaining previous transaction logs for a long period of time. In contrast to noncircular logging in which transaction logs continue to accumulate until a full backup (or the maximum number of files permitted is hit), only a few transaction logs are maintained at any one time—usually four. Circular logging continually overwrites the oldest logfile when a new transaction logfile is needed, unless the size of the transactions in the logfile exceed 5 MB. If that happens, a new file is created until the checkpoint can be advanced. After the checkpoint has been advanced, these extra logfiles are generally not used and are deleted during the next backup.

> **Warning ☠**
>
> It should be reiterated that this type of configuration should not be used for any Exchange Server in which you are storing user data. This type of arrangement works best for frontend-pass through type servers where no mailbox or public folders reside or for NNTP servers where it is expected that the data is short-lived. Basically, use this configuration only in places where recovery isn't important to you.

Other Files

In the interest of completeness, you may see some other files in your Exchange directory from time to time. These are temporary files and are a normal function of the Exchange Server.

E00tmp.log

: This file is created by ESE for temporary storage as the transaction logfile is being renamed and a new transaction logfile is generated.

tmp.edb

: This file is created when the Exchange Server needs temporary storage during maintenance times, for temporary tables, sorting, or index creating.

stf files

: These temporary files are created by the IMAIL process during content conversion.

IFS files

These temporary files are created by ExIFS and contain cached directory lists of the ExIFS.

Figure 20-5 is an excellent representation of file sizes for a low-volume Exchange Server. Here we have a sampling of almost everything covered. Notice that the transaction logfiles have incremented since the last full backup (*E0000335.log* and *E0000336.log*) and that the file sizes are consistent with what was discussed. The checkpoint file is 8 K, and the transaction and reserve logfiles are all 5 MB.

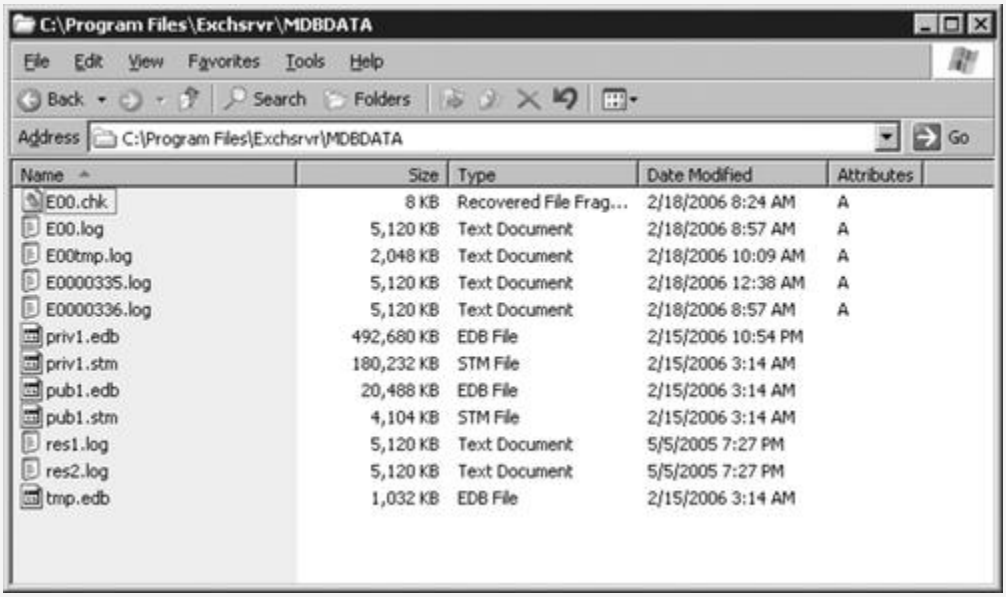


Figure 20-5. Transaction logs in Exchange Server

Single Instance Storage

Exchange employs a great storage savings feature called *single instance storage*. When a message is sent to multiple mailboxes that reside in the same store, only one instance of the message is stored in one mailbox; other mailboxes will contain a pointer to the stored message. This feature is maintained on a per-store basis. It is important to remember that if the message is sent to mailboxes that are in different stores, storage groups, or servers, it is stored

once per store, storage group, and server. For example, if a message is sent to two mailboxes on two servers, it is stored twice. If it is sent to two mailboxes that are in different stores or storage groups on the same server, it is stored twice. But if it is sent to two mailboxes that are part of the same store, it is stored only once.

Automatic Database Maintenance

Automatic database maintenance occurs every night at intervals designated in the Database tab. While a great deal of maintenance can go on during this time, we do not cover it in detail since we are only interested in how it applies to, or interacts with, backups. The important thing to note about automatic database maintenance is its scheduling. Automatic maintenance can be CPU-intensive and therefore should be scheduled at a time when backups aren't running. Additionally, this maintenance window should occur when the fewest users are online. Finally, if there are multiple storage groups on the same server, stagger the time in which each runs so as to avoid overlap.

Storage Limits

Exchange 2003 Standard Edition can have one storage group that contains one mailbox store and one public folder store. The maximum size of a store prior to Exchange 2000 and 2003 Service Pack 2 is 16 GB. In Service Pack 2, this limit is 75 GB.

With Exchange Server 2003 Enterprise Edition, you can have up to four storage groups, each of which can contain up to five databases, either mailbox stores or public folder stores. The maximum size of each store is 16 TB.

Backup

Now that we have examined the various elements of Exchange architecture, let’s take a look at how to back it up, starting with your backup strategy.

Backup Strategy

Your backup strategy is the starting point in the success or failure of your eventual restores. A good Exchange system administrator shouldn’t be thinking about “if” some piece of hardware or software will fail on his Exchange Server, but “when.” Taking proactive steps to ensure your backups occur at regular intervals and that the backups you make contain useful and necessary data can make the restore a breeze and help make you the hero of your organization.

When figuring out your backup strategy, keep several things in mind. Does your organization have a predefined disaster recovery plan (DRP) to which you might be bound? For example, if you do business with certain government agencies, you may have strict rules governing what must be backed up and how often. Does your group have a service-level agreement (SLA) with other groups in the company? This SLA might also dictate that you have certain response times to mailbox recovery or just generally provide guidelines you must follow for your industry.

While considering what type of backup strategy to implement, it will also be necessary to think of the technical side. How long will the backups take? How much storage will I need to perform any of the different types of backups? Should I do full backups all the time or will differential or incremental be sufficient?

Backup Types

Backup types are specified using the Default Backup Type drop-down on the Backup Type tab under the Options setting in `ntbackup` as shown in Figure 20-6. Here you can specify the type of backup you want performed. Each type is described in the following subsections. In 2000, simply run the Backup wizard (Start→Run, and type `Backup`), and specify the type of backup on the "Type of Backup" page. For the remainder of the chapter, screenshots are specific to Exchange Server 2003. Almost all the same options are available in 2000; simply run the wizard to configure your backup, and set these parameters.

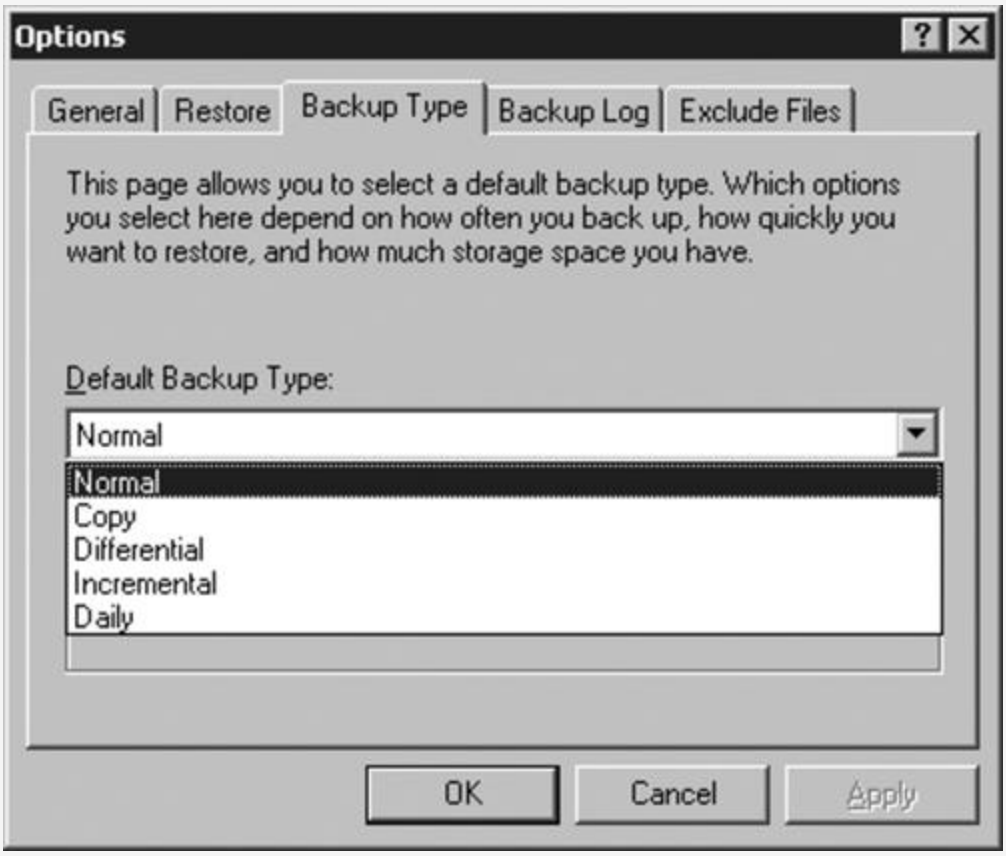


Figure 20-6. Backup types

Normal

A *normal* or *full* backup is the safest bet when backing up your Exchange Server. This backup type provides a full backup of all the information you select. It is important to do a full backup regularly so that the transaction logfiles are purged. Otherwise, it is possible to hit the maximum limit of logfiles. While this number is high, currently 1,048,540 for 2003, in a high-volume environment it is conceivable to reach this limit quite easily. Keep in mind that while full backups are supposed to purge old transaction logs, there are times when this is not the case. Currently, if a database within the same storage group is dismounted when you select it for backup, the transaction logs are not purged. Additionally, if you don't select all the databases within a particular storage group, only the transaction logs that have the oldest full backup PIT are truncated.

Copy

A *copy* backup is a backup in which all the Exchange Server files are copied onto the backup medium without purging the transaction logs or updating the last backup date. The main advantage of this type of backup is that it can be performed without affecting any of the other backups.

Incremental

With Exchange Server 2003, *incremental* backups copy only the logfiles from the last full or incremental backup to the backup medium. The advantage here is that incremental backups are much faster than full backups. However, it takes all the incremental backups plus the last full backup to bring Exchange to it most recent state.

Differential

Differential backups are similar to incremental backups because they both back up only the logfiles. However, unlike incremental backups, the transaction logs are not purged during a differential backup. This makes restoring from a differential easier than restoring from an incremental because you need only the last full backup and the last differential backup to make the Exchange Server current. The downside to this method is that since the transaction logs are not purged, it is necessary to keep a watchful eye on the number of logfiles and implement one of the previous two backup strategies to make sure the logfiles are purged at some point. For smaller organizations this may not be an issue, but if you are managing a high-volume Exchange Server, this can be problematic.

Daily

Do not confuse *daily* backup with the generic terminology of a daily backup. In this context, it refers to a copy backup that backs up only the current day's data. Like the copy backup, it does not purge the transaction log or update the last backup date.

Determining What to Back Up

A fully successful backup of your Exchange 2003 Server may entail more than just the databases and transaction logs. A host of additional information may need to be restored to bring your Exchange Server to a fully operational state.

Exchange-specific

It goes without saying that in order to restore a failed Exchange Server you'll need all the Exchange components

restored. These include Exchange databases, stores, storage groups, transaction logs, and checkpoint files.

Other Exchange pieces you may wish to back up include connector information, full-text index, message tracking logs, site replication service, Exchange cluster data, SRS database, the application software itself, third-party software such as Exchange-specific virus scanners, and any scripts you've created for ease of administration.

Windows-specific

While it is important to back up your Exchange Server, it is just as important to back up the server that Exchange lives on. Depending on your organization's particular setup, you may have more than just Exchange data on this server. Often Exchange is intermixed with Active Directory or, as in the case of Small Business Server, all your primary functions live on the same server.

Chapter 11 provides you with the details necessary to recover a failed Windows server, but it is worth mentioning that it is always a good idea to back up the following Windows server components in addition to your Exchange data: Registry, IIS metabase, Active Directory, operating system files, system state, boot files, boot partition, partition information, COM+ class registration database, *SYSVOL* directory, DNS information, cluster service, and certificate service database including the certificates themselves.

What not to back up when backing up Windows

Take special precautions when backing up your server that you have a backup application that is Exchange-aware. If your backup program doesn't have links to the Exchange API that coordinate the backup with Exchange, you run the

risk of corrupting your databases and will not have an effective backup of your Exchange Server. In order to prevent this, exclude the following components from your server backup if it isn't Exchange-aware: database and logfiles (anything in the *MDBDATA* folder), and installable filesystem drives (IFSs).

Backup Methods

The days of having to bring all databases to a quiesced state in order to back them up are long gone. While not new in Exchange Server 2003, online backups are much improved and safer than previous versions. As with 2000, there is no reason in 2003 to stop the server or dismount the stores in order to get a good, complete backup. In fact, with Exchange 2003, if you don't do regular online backups, you could be doing your server more harm than good because the transaction logfiles are not being purged. Additionally, many consistency checks are done for both 2000 and 2003 during the backup; if these checks fail during the backup, the backup will terminate.

Online backups

Online (a.k.a. hot) backups are now the typical method for backing up your Exchange Server. As the name implies, online backups can be performed while the database is operational, avoiding any downtime that may impact your users. Additionally, online backups provide a cyclical redundancy check (CRC) as the data is backed up for extra verification of the data; the offline mode does not. Backups performed online also clear out the ever-increasing number of transaction logfiles.

Offline backups

While online backups are the preferred way to back up your Exchange Server, *offline* backups are not without their place. The main disadvantage of offline backups is the fact that the storage group must be dismounted, which means that the database is unavailable to users during this time. Also, because of the nature of this backup, no transaction logs are purged, and the last full backup flag is not set. The only way to rectify this situation is to perform one of the backup types that purge the files. One advantage of offline backups is that they can include all Exchange configuration data, such as connector information. Other benefits are the ease with which these backups can be performed and their speed. Backups of this type can be scheduled through `ntbackup` or, more simply, the *.edb* and *.stm* files can be copied with system commands such as `copy` or `xcopy`.

Streaming backups

Microsoft Exchange Server 2003 provides an API that can provide access to *streaming backups*. This method provides continual backups that stream to the backup medium. This ensures that the backup is always up to date.

Shadow copy backups

Microsoft Exchange 2003 Service Pack 1 supports a feature called Volume Shadow Copy Service when running on Windows Server 2003. This technology allows for point-in-time snapshots to be taken of the Exchange databases. These snapshots can be one of two different types, clone or copy-on-write. As the name suggests, the *clone* is just that, a complete copy of the original volume. The *copy-on-write* method stores only the changes made to the original volume since the last snapshot. Both methods can be implemented either in software or hardware. However, only

those backups made to hardware are transportable, or able to be moved to other servers.

In order to use Volume Shadow Copy Service, your backup software, the application you are trying to back up (in this case, Exchange), and your backup device must support it. There are three capability levels for the Volume Shadow Copy Service:

Requestor

: Generally an additional piece of software, such as a backup application, that can use the shadow copy feature.

Writer

: The component that enables snapshots to be taken, generally application-specific, such as a writer for Exchange or SQL Server.

Provider

: Software or hardware that actually writes to the hardware.

The default behavior for `ntbackup` is to operate as a Volume Shadow Copy Service requestor. However, the `ntbackup` requestor uses only the copy-on-write method. In order to back up using the other method, a third-party application that ties into the Volume Shadow Copy Service is necessary.

Because of the nature of these snapshots, and the fact that data is constantly streaming to them, you need a lot of storage space. For that reason, a NAS or a SAN device is the perfect storage location for shadow copies. Some NAS vendors have even implemented this feature directly into their hardware, making backups and restore incredibly easy.

Verifying backups

As important as it is to make backups, it is just as important to verify them. All of your hard work making a backup schedule and implementing it is completely and utterly useless unless you verify that the backups you are making can be restored.

The first step in verifying your backup is to review the backup log after the backup has completed. This gives you an overall status but could lack some specific details. For specific details, review the application log in the Event Log. This often provides a great deal more information that the report. There is also a verify option for `ntbackup` so that you can have it verify the backup. This is a good plan in addition to the other steps, but don't rely on this method alone.

For the ultimate in verification, consider doing a restore. A restore to a test server has many benefits besides just proving that your backups are fine; it also initiates you into the restore procedure so you can prepare for a real emergency. In addition, it gives you a chance to walk through the steps involved in a non-job-threatening situation, so you can calmly and collectedly take notes and get your restore procedure down. That way, when the real emergency occurs, you are a hero.

Using ntbackup to Back Up

`ntbackup` can be used to back up Exchange live because Microsoft has connected the two using Exchange's API. This is the only way to perform automated backups of Exchange without purchasing a third-party tool.

Making a Basic Backup

To start the backup utility, go to Start→Run, and type `ntbackup`. When you initially start `ntbackup`, it starts in wizard mode as shown in Figure 20-7. If you're running Exchange 2000 on Windows Server 2003 or later, the procedure is the same. If you run a version of Exchange on Windows Server prior to 2003, the backup utility is called `backup` and can be executed by going to Start→Run, and typing `Backup`.



Figure 20-7. ntbackup: wizard mode

If you are a more advanced user and prefer to not use the wizard but instead start in advanced mode, you can uncheck the “Always start in wizard mode checkbox” and then click the Advanced Mode link. To switch back to wizard mode, select Tools→“Switch to wizard mode.”

Click the Advanced Mode link to display the choices shown in Figure 20-8. From here you can easily work in wizard mode by selecting the Backup wizard, or you can select the Backup tab to manage the backup. In this example, we perform a backup without the wizard and proceed to the Backup tab.

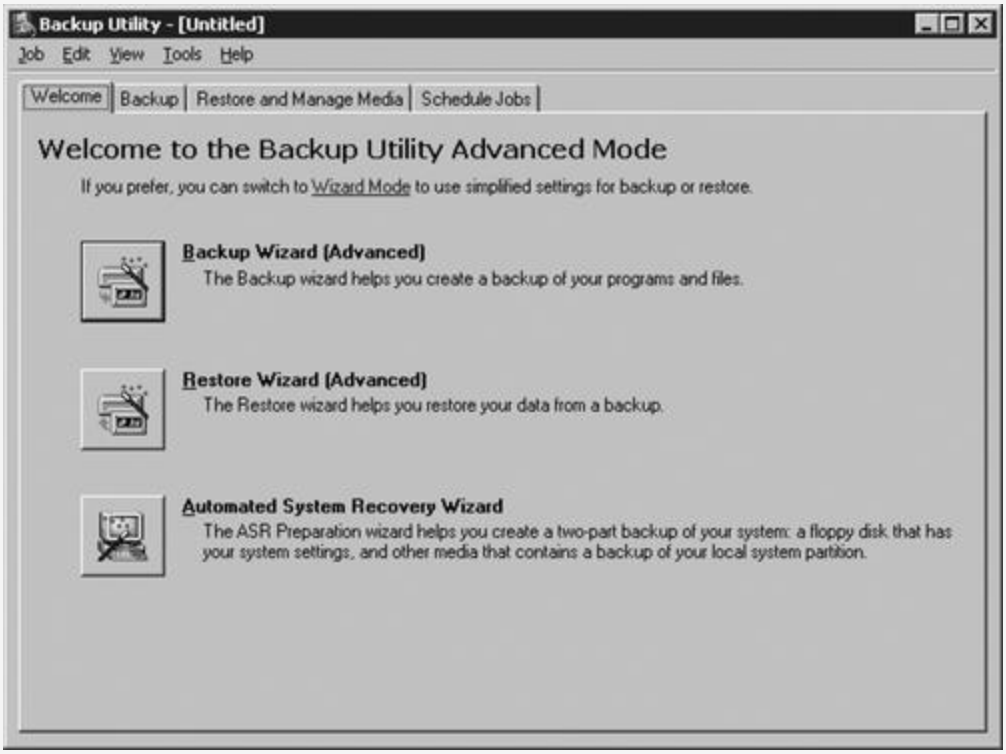


Figure 20-8. Backup wizard main screen

Which components are installed on the Exchange Server dictates which components need to be backed up. To back up all the storage groups on the server, select the checkbox next to Microsoft Information Store. To back up a specific storage group, expand the Microsoft Information Store, and select the storage group.

In Figure 20-9, we have specified the Information Store for the Exchange Server and that the backup location should be *E:\Exchange\Backup.bkf*. This will be a file-based backup to externally attached storage, such as a NAS or SAN. Other options include backup directly to tape. On the right pane, we specified both the mailbox store and the public folder store.

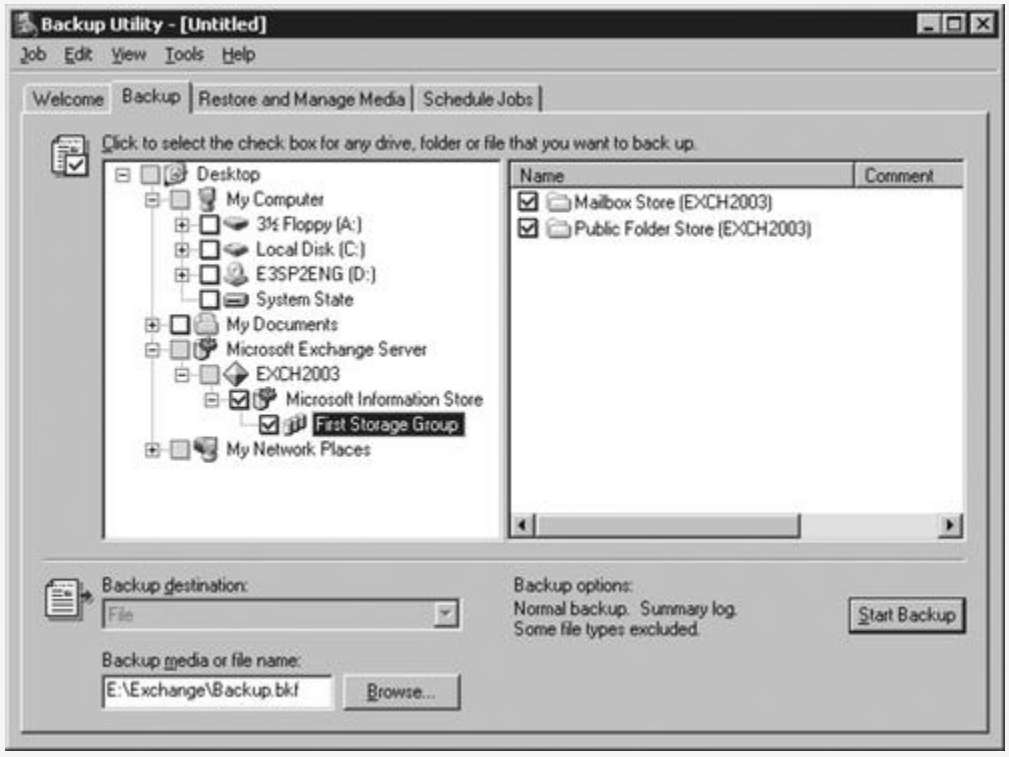


Figure 20-9. Storage group selection for backup

After selecting the components to back up, click the Start Backup button to display a window like the one shown in Figure 20-10.

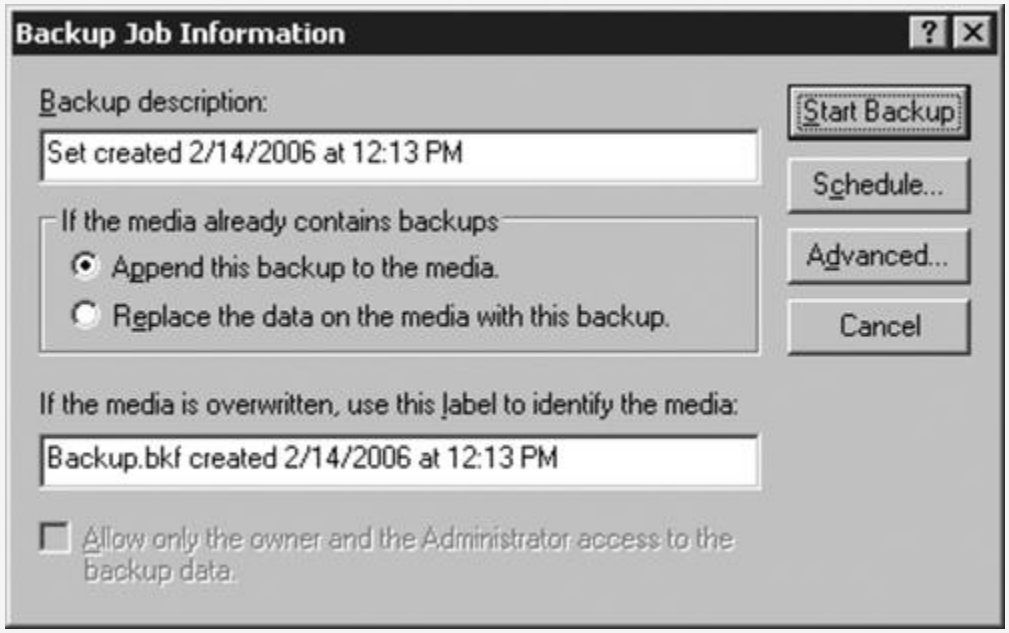


Figure 20-10. Backup-job information

From this screen, you can (and should) enter a detailed description as well as specify advanced backup options or schedule this backup instead of running it immediately.

If you click the Advanced button, you are presented with the dialog box in Figure 20-11, in which you can change some of the default options or the type of backup you'd like. It is always a good idea to verify the backup after it is complete, so this checkbox is selected. If you change the type of backup on this screen, be careful to select a backup that clears the transaction logs and marks the backup flag.

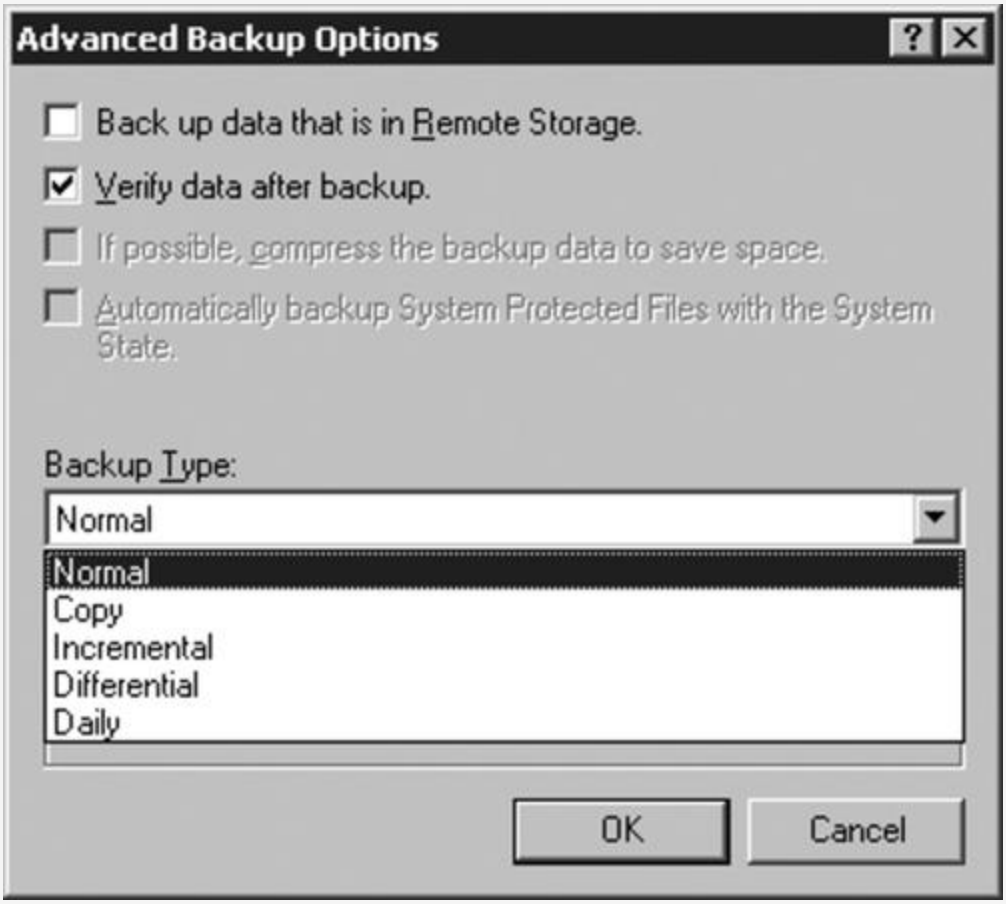


Figure 20-11. Advanced backup options

After selecting your options, click OK to return to the main menu, and then click Start Backup to begin. You'll see a status window like that shown in Figure 20-12.

The next step, covered in the following section, is to verify the backup.

Verifying the Backup

After the backup has completed, make sure it was successful. This can be verified by first checking that the backup status is Completed in the Backup Progress window, as shown in Figure 20-12.

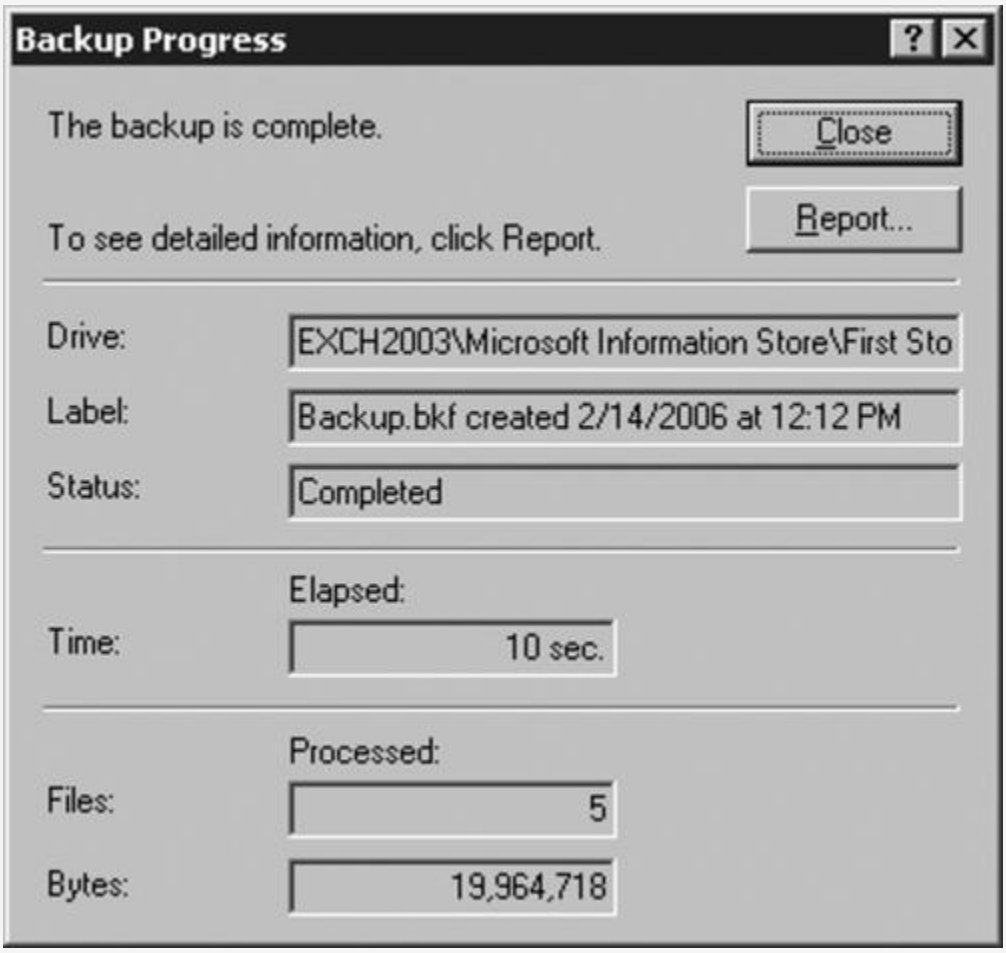


Figure 20-12. Backup progress

As tedious as it might sound, click the Report button to view the backup report to verify that even though the backup was successful, there were no errors. You will see a report similar to the one shown in Figure 20-13.



Figure 20-13. Backup verification by viewing the backup logfile

The default report is only a summary. If there are errors, and you need more details, rerun the backup with detailed logging enabled. Because all backup jobs on this server append to this logfile, it may be necessary to scroll through a considerable amount of information to find the offending error.

If you checked the “Verify data after backup” checkbox, check the report to confirm that the backup was successful. To do this, look for a line that starts with the word *different*. If it is anything other than zero, there might be problems, but nonconformity between the files does not always

indicate a problem; you must investigate each case individually.

While these methods report most errors, the authoritative place to check is the *application event log* (see Figure 20-14). It is important to scan the application log regularly for backup errors because not all errors are logged to the `ntbackup` logs. This job can be made easier by looking between event ID 8000 and 8009, which indicate the start and end of the backup, respectively. Pay particular attention to any sources that are `ntbackup` or ESE.

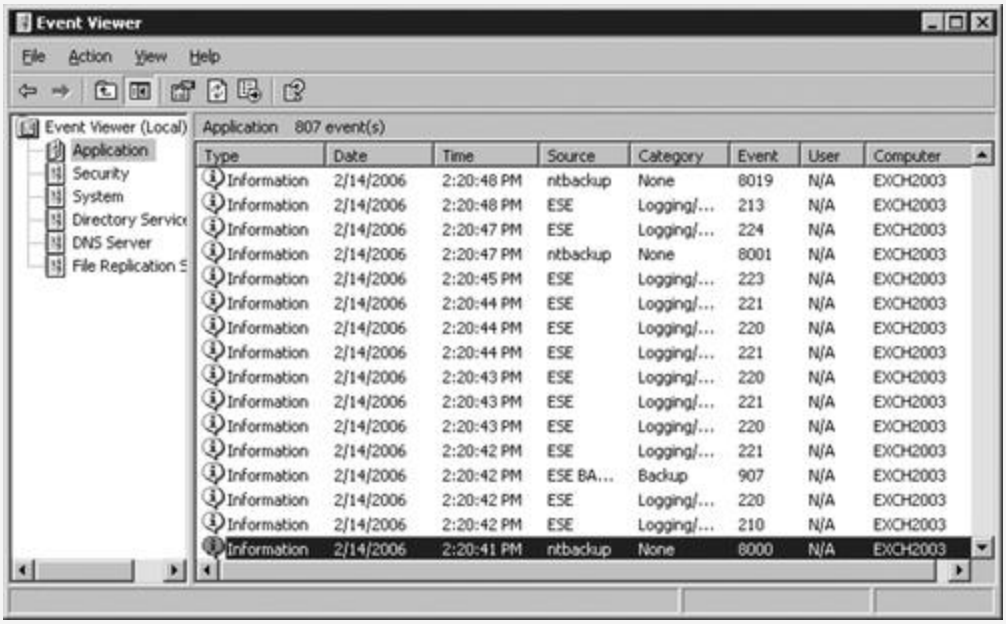


Figure 20-14. Backup status in Event Viewer

If you don’t find any errors, there is a good chance that your backup was successful. However, if you want the ultimate verification, you should do a test restore, covered next.

Restore

This entire backup is pretty useless unless you can restore after a meltdown. This section will help you in that arena.

Repair or Restore?

One of the first questions you'll need to ask yourself after a disaster is if it is worthwhile to try to repair the existing Exchange Server or restore from backup. New tools and technologies make repairing much easier than in previous versions, especially at the individual mailbox level.

Common Tasks for Repair or Restore

Repairing and restoring have some tasks in common. For example, before you jump in and start looking at Exchange, make sure your foundation is solid. In this case, your foundation is the Windows Server on which your Exchange Server resides. Running a few simple verification exercises can save you hours down the road. Many times I have seen administrators spend hours repairing or restoring their Exchange Server, proudly receiving kudos and accolades from their coworkers, only to see the server crash and burn soon afterward because the foundation wasn't solid.

The first thing to do before you start repairing or restoring is to ensure the server on which Exchange is running is operating correctly. This includes things such as:

Hard disk media check

> Use `chkdsk` or some other utility provided by your disk vendor to verify the integrity of the actual media.

Windows Server file verification

Use `chkdsk` to verify the filesystem and the filesystem metadata.

Last known good configuration

When booting Windows, try selecting the last known good configuration.

Windows Recovery console

This option is available in Windows Server 2000 and beyond and is accessed when booting from the operating system CD.

Restoring with backup sets

Restoring with backup sets is, of course, always an option. However, it can be quite tedious and take a significant amount of time. This method entails first building a replacement server and then restoring previous backup sets to create a functioning server.

Automated System Recovery (ASR)

ASR provides a convenient and automated way to restore your Windows 2003 server. Using the ASR backup floppy in conjunction with booting from the Windows Server 2003 CD, you can fully restore your server to its last ASR backup state.

Obvious problems

While it may seem trivial, ensure that Exchange hasn't shut itself down due to the disk filling up and running out of space.

Exchange Repair

After you've ensured that your Windows server isn't the problem, or if it was, that you've now repaired it, it's time to move on to the Exchange Server. Recall those two

possibilities: repair the existing installation or restore from backup. Whichever option you choose, back up the server first, no matter what state it is in. This ensures that if your repair or restore efforts are unsuccessful, you at least have the latest point-in-time backup, even if it is corrupt. If things become so fouled up trying to restore or repair, at least you know you can get back to where you were right after the failure.

Repairing Exchange databases

Included with Exchange are several command-line tools that can help you repair corrupted or damaged database files and message stores. In a 2003 standard installation, these tools are found in *C:\Program Files\Exchsrvr\bin* (substitute the appropriate drive letter and path depending on your particular installation). `ESEUtil` is a tool that can check the integrity of your database or repair a damaged one. The `ISINTEG` tool tests an offline store for integrity errors and can also repair those errors. In 2000, `ISINTEG` is found in *C:\Program Files\Exchsrvr\bin*, and `ESEUtil` is located on the Exchange 2000 CD in the *D:\Support\Utils* folder and must be installed (where *D:* is the appropriate drive letter for your CD drive).

These are very powerful tools and offer a great deal of flexibility and options to help restore or recover an Exchange database. In fact, in some instances the only options for recovery or repair are these command-line utilities. More information about these tools can be found online at http://support.microsoft.com by entering either `eseutil` or `isinteg` as the search term.

Repairing by reinstalling

One option for getting your failed Exchange Server operational is to reinstall on top of an existing installation. This generally works for instances in which some of the executables or DLLs necessary for Exchange to function have become corrupt or were accidentally deleted, or in which registry entries have become corrupt.

Of course, if you are taking this route, the Exchange Server processes are shut down during the reinstall and inaccessible to users. And it is important to remember that you must bring any software you install to the same patch level that was installed when the backup was created.

If you choose this method, use the `/DisasterRecovery` switch to tell the installer not to make changes to Active Directory. This keeps the installer from registering a new Exchange Server in Active Directory. Keep in mind that the Exchange Server configuration must still be in Active Directory. If you manually deleted the information from Active Directory (or the server has been down so long that it has been purged), you'll need to take extra steps in order to repair.

Exchange Restore

Assuming the rebuild and repair options didn’t work, here is an overview of how to restore Exchange.

Overview

The following synopsis describes the overall restore process:

- The stores are dismounted.
- Database files from the backup medium are copied to the original location on the Exchange Server, and if they exist, they overwrite the *.edb* and *.stm* files.
- The transaction logfiles are copied to a specified temporary location.
- Transactions are replayed.
- The *restore.env* file that contains information about logfiles (such as what storage group they belong to, database paths, logfile ranges, and so on) is copied.

Restoring Exchange Mailbox or Public Folder Stores

You can choose from several methods when restoring your Exchange stores. The basic procedure is similar for most methods, as outlined earlier.

Online database restore

An *online database restore* is almost always the easiest type of restore because it is the primary focus of most backup

applications. With `ntbackup`, it is as easy as dismounting the store you wish to restore, running the aforementioned program, and selecting which backup set to restore. `ntbackup` takes care of the rest. A detailed walkthrough of a basic online restore can be found later in this section.

Incremental and differential backups

From the backup section, we learned that incremental and differential backups saved a great deal of time during the backup process. Now comes the time when those savings could potentially be lost, when we want something the fastest, during the restore. The issue is that now that you are ready to restore, you must restore your last full backup, then proceed to restore all the incremental or differential backups to the same temporary location. Then, when the last backup medium is loaded, you specify that this is the last backup, and the replay of the transaction logs starts. This places you at the mercy of your backup medium. If you are backing up to tape, you are likely to encounter more problems than if you backed up to a SAN or NAS.

If you did not check the Last Restore Set checkbox on the Restore Database Store screen, or you are copying files from other locations, you have to do a manual hard recovery in order to play back the transactions.

Hard recovery versus soft recovery

A *hard recovery* is the act of replaying the transaction logs after the restore of a database from an online backup. After the hard recovery has successfully completed, the database is said to be in an inconsistent or unsynchronized state. One major difference between hard and soft recovery is that with a hard recovery, the checkpoint file is not used; the *restore.env* file is used instead. This file specifies the

range of transaction logfiles that must be present in the temporary folder.

A *soft recovery* is the act of replaying the transaction logs into an offline file copy backup. This process assumes that all the files necessary are all still available and not corrupt. The checkpoint file is used to determine from which point to start the replay. If no checkpoint file exists, replay begins at the oldest transaction logfile.

Offline Database Restore

An *offline restore* is a restore in which you simply copy the files from backup media into the proper location on the Exchange Server. Before you begin an offline restore, you need to decide if you want a point-in-time restore or a rollforward restore:

Point-in-time restore

: No logfiles are replayed, and all of the data created after the last backup is lost.

Rollforward restore

: Available logfiles are replayed to create the restore. If all the logfiles are available, it is possible to restore the database fully. If circular logging is enabled, only PIT restore is available.

Recovery Storage Group

Recovery Storage Groups (RSGs) are a great feature of Exchange Server 2003. They allow you to mount a second copy of a mailbox database on the same server while the primary database is still online. This can be a great time saver for the Exchange administrator because in prior

versions of Exchange it was necessary to build an entire replacement Exchange Server to mount the databases. After the RSG is created, you can restore your backups to this database with no impact on the current system or its users. After the restore, you can use the Mailbox Merge wizard (ExMerge) or the Recover Mailbox Data feature to merge the information back into the primary storage group.

Recovery Storage Groups should not be viewed as your main savior from disaster. Instead, they are a supplemental recovery method that complements existing recovery methods. You should keep in mind that RSGs are primarily designed to recover individual mailboxes and do not work with public folders.

Example of RSG

Run Exchange System Manager from Start→All Programs→Microsoft Exchange→System Manager. Right-click on the server where you wish to create the RSG, and select New→Recovery Storage Group, as demonstrated in Figure 20-15.

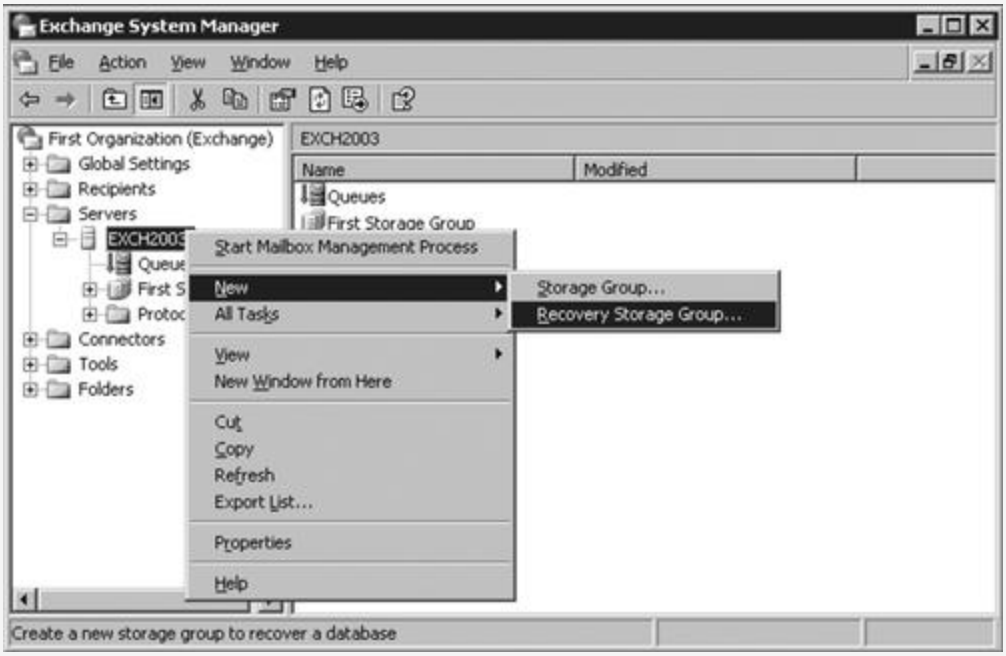


Figure 20-15. Location of Recovery Storage Group in System Manager

The default locations for the logfiles and system path are shown in Figure 20-16. These should be adjusted for your specific configuration. After the values have been corrected and verified, click OK to create the storage group.

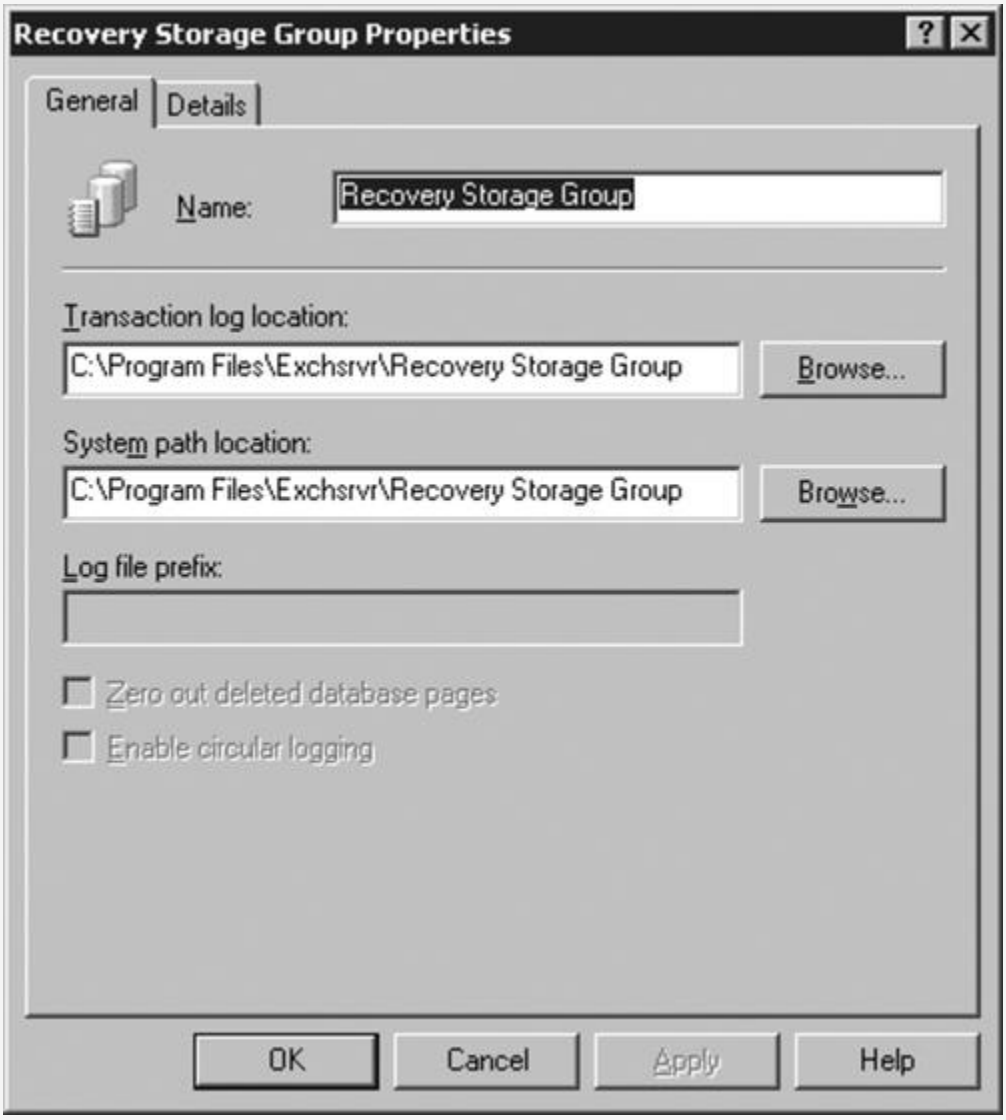


Figure 20-16. Properties window for Recovery Storage Groups

Now your storage groups should look similar to those in Figure 20-17 if you have a standard Exchange installation. There are now two storage groups, the First Storage Group and the Recovery Storage Group. It is worth noting that no email can be delivered to the Recovery Storage Group.

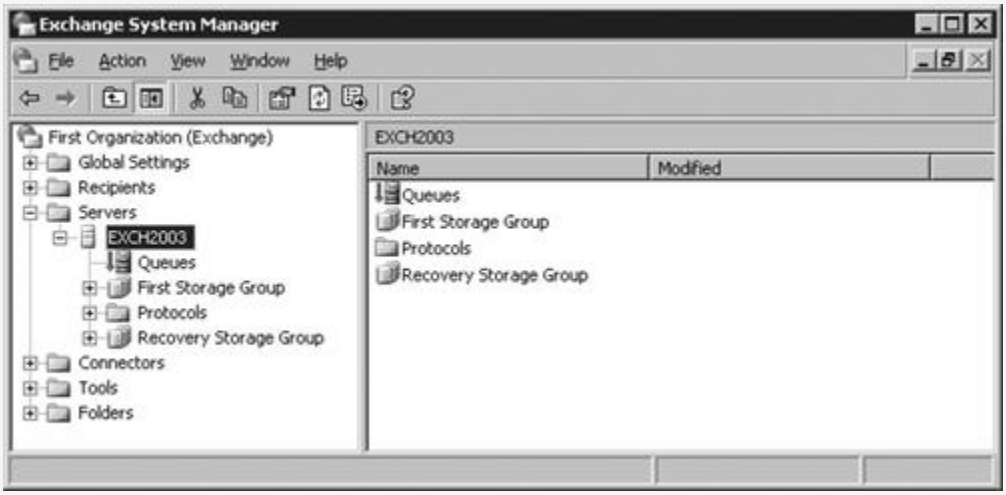


Figure 20-17. Recovery Storage Group in System Manager

Right-click the Recovery Storage Group, and select "Add Database to Recover" as shown in Figure 20-18.

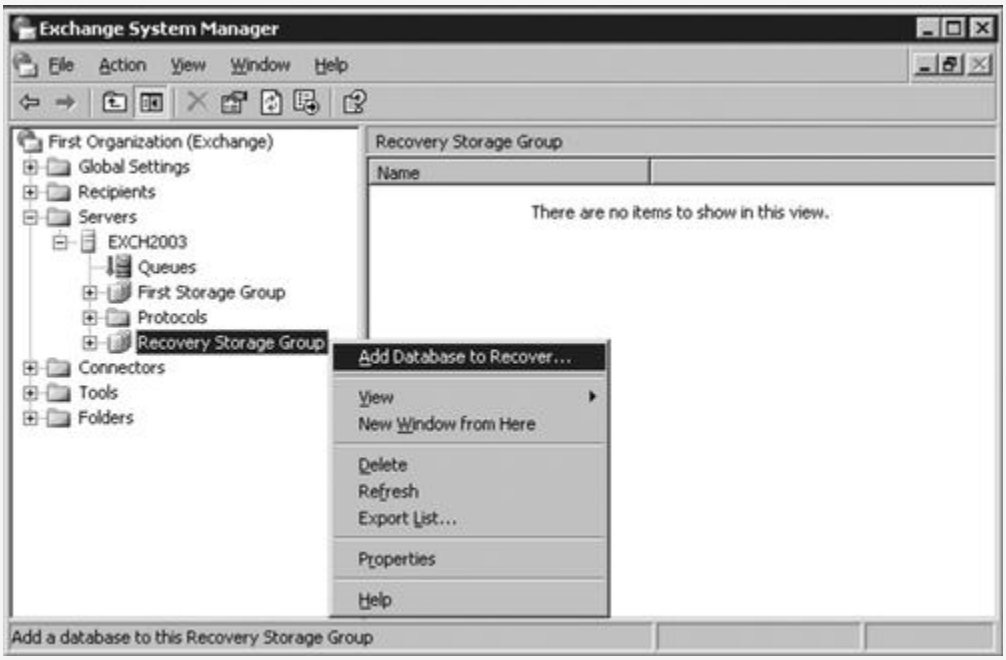


Figure 20-18. "Add Database to Recover" option for Recovery Storage Groups

The dialog box in Figure 20-19 is then displayed. Select the backup you wish to restore to the RSG, and click OK.

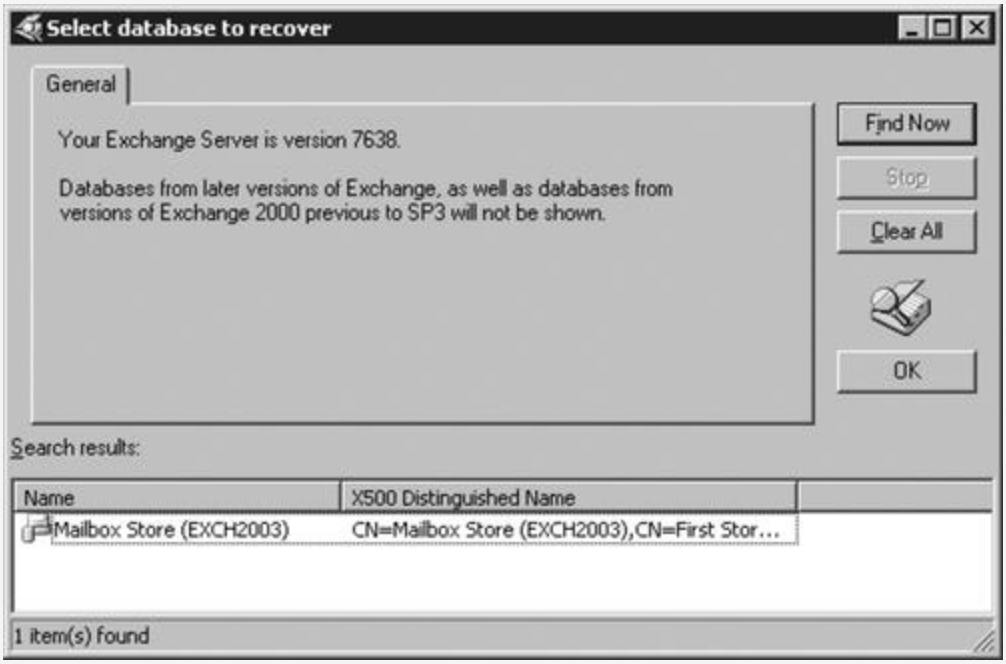


Figure 20-19. Database recovery selection

On the Mailbox Store Properties dialog (Figure 20-20), go to the Database tab and select a location to restore the *.edb* and *.stm* files. After you've adjusted the location or are happy with the current location, click OK.

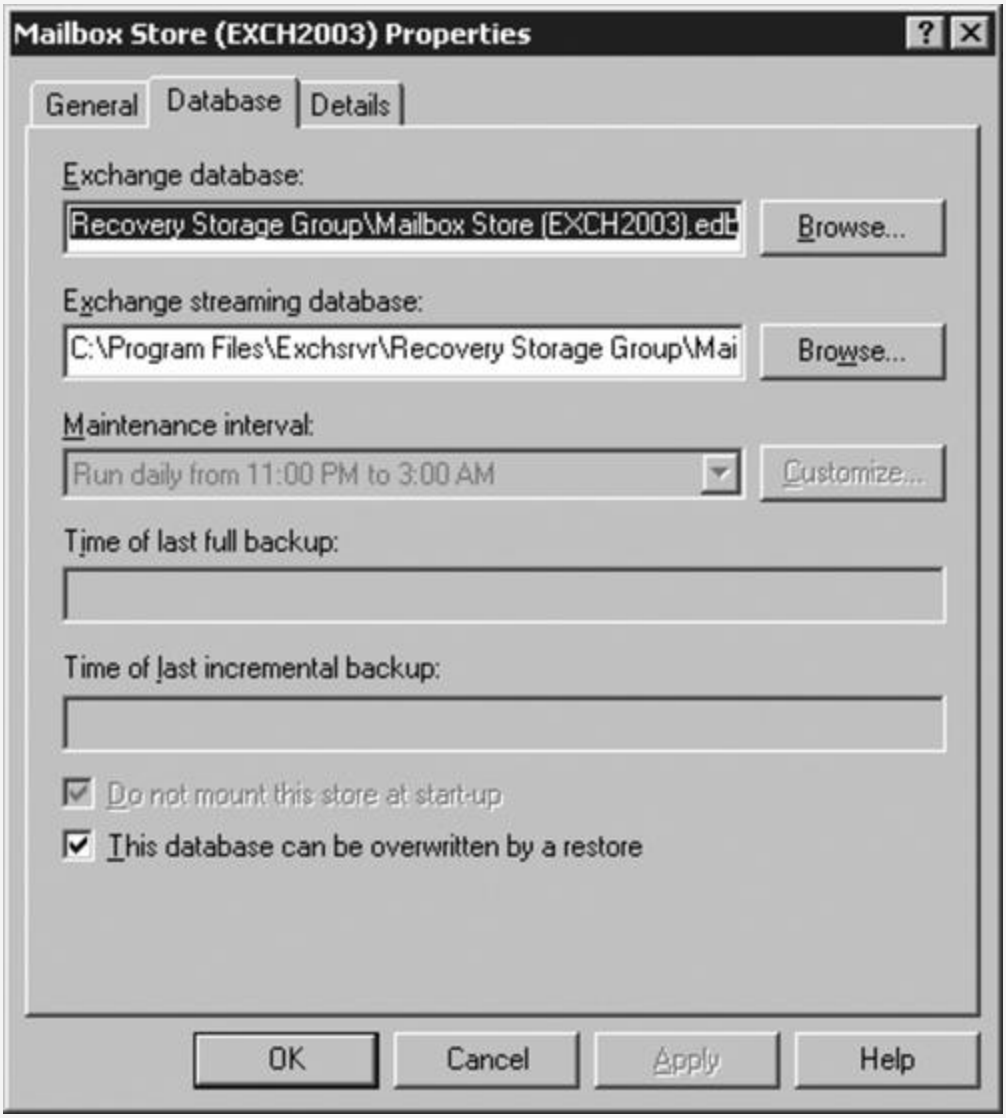


Figure 20-20. Mailbox Store Properties dialog

Your mailbox store is now restored into the RSG. It will always restore to an unmounted state, so you must manually mount the store.

Now you can run your restore as usual. If the restore process finds an RSG, it automatically uses this storage group rather than the default.

When to use RSG: Dial-tone restore

A perfect example of when to use the RSG involves a method called *dial-tone database restore*. The premise is to restore the mailboxes as soon as possible, most likely without messages (that is, get the dial tone back). This enables users to send and receive new messages immediately. From this point, you can take a more leisurely approach restoring the mailboxes. While this doesn't give the users their data immediately, it does keep operations at the company from grinding to a halt due to unavailability of email.

This is especially useful for companies that use email as their primary method of communication. If the Exchange Server crashes due to a hardware failure and needs a full restore, and the restore includes large amounts of data, it may take hours and hours. In this scenario, you give the users the ability to send and receive email immediately while you restore the backups.

Overlooked (and Often Easy) Restore Methods

Usually when something bad happens, the administrator immediately has a flash of terror as he thinks about having to do a full restore, at 4:00 a.m., for the president of the company. No small task indeed! However, some new and easy methods for mailbox and message recovery are often overlooked.

Restoring deleted mailboxes

Exchange Server has a feature in which you can delete a user's account and the mailbox can remain for a predefined number of days, disconnected from any user account. This allows a grace period from the time the user account is

deleted until the mailbox is deleted from the store. Figure 20-21 is an example of a user called Temp User who has been deleted. Notice that the mailbox is still present in Exchange System Manager, but its icon has a little dot with an X in it that indicates that it does not have an associated user account.

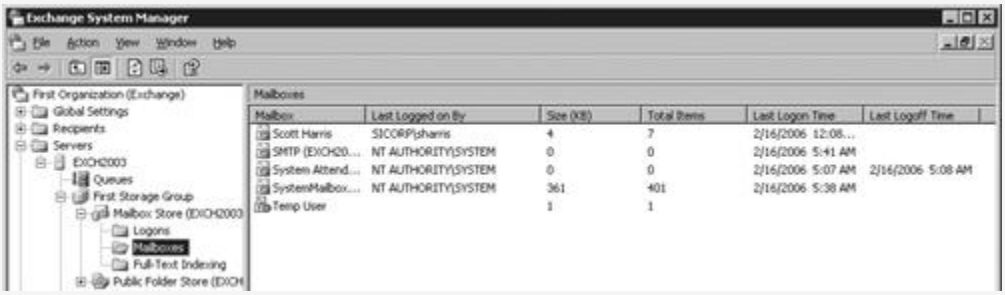


Figure 20-21. Delete-user example

From this point, there are two ways to reconnect the mailbox to a user account. In each case, you must create a user account without an associated mailbox.

If you right-click on the Temp User mailbox, you can select the Reconnect option. This displays the object searcher where you can find a user account to try and reconnect. After a user account without a mailbox is found, select it, and click OK. The mailbox is now connected with that user account.

The second option is to run the Mailbox Recovery Center. The principles with this method are the same, but you need to mount a store with unattached user accounts first. After that point, the reconnection is virtually identical. This ability to mount a different store allows the Mailbox Recovery Center to import mailboxes more easily from backups of different stores. Keep in mind that Active Directory permissions hold here, so reconnection may not always work due to permissions issues.

Restoring deleted items

The ability to restore deleted items is one of Exchange's greatest features because it takes the burden of the restore process off the administrator's hands and places it directly in the user's. If you are an Outlook user and need to restore something long since deleted and not simply moved to Delete Items, you can do this with the Recovery Deleted Items option from within Outlook. Simply click on your Deleted Items mail folder, and then select Recover Deleted Items from the Tools menu. A window such as the one in Figure 20-22 is displayed where you can undelete the message.

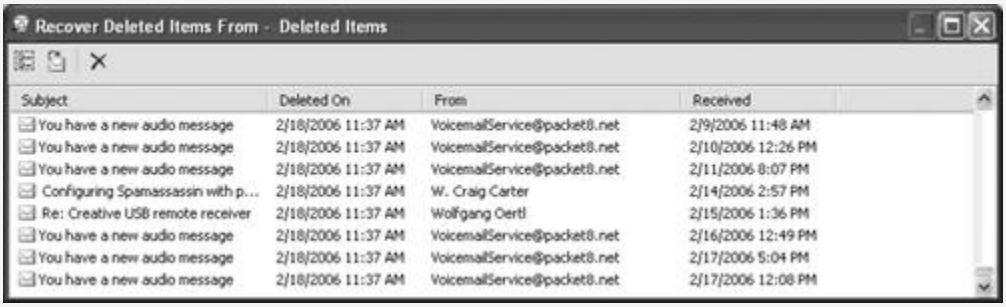


Figure 20-22. Recovering deleted items from Outlook

For this option to work, the Keep Deleted Items setting must be enabled on the mailbox store. These options are set in the Limits tab of the store's Property option in Exchange Server Manager, as seen in Figure 20-23. This method also works for public folder stores.

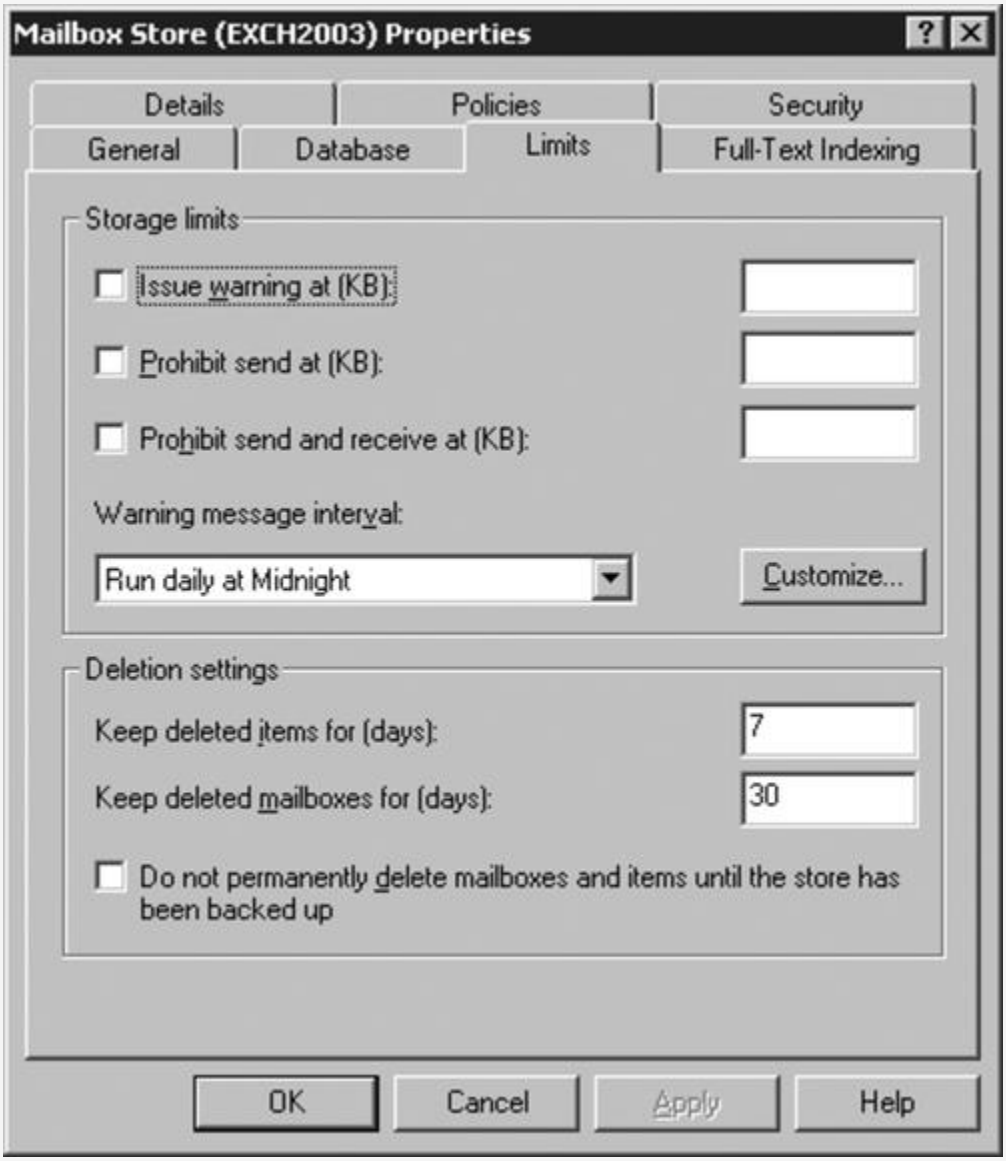


Figure 20-23. Storage limits for deleted items

Using ntbackup to Restore

`ntbackup` can also be your friend when it comes time to restore.

Performing a basic restore

Before you begin, you need to take the stores you wish to restore offline and allow them to be overwritten by the backup. To do this, open up Exchange System Manager from Start→All Programs→Microsoft Exchange→System Manager. Expand Servers, then the server you plan to restore, then the storage group, and here you will find the stores. Right-click on the stores you are going to restore, and select Dismount Store (Figure 20-24). Because this is a basic restore, remember that dismounting the store makes it inaccessible to users. As previously mentioned, the procedure is very similar for 2000; simply use the backup wizard by selecting Start→Run, and typing `Backup`.

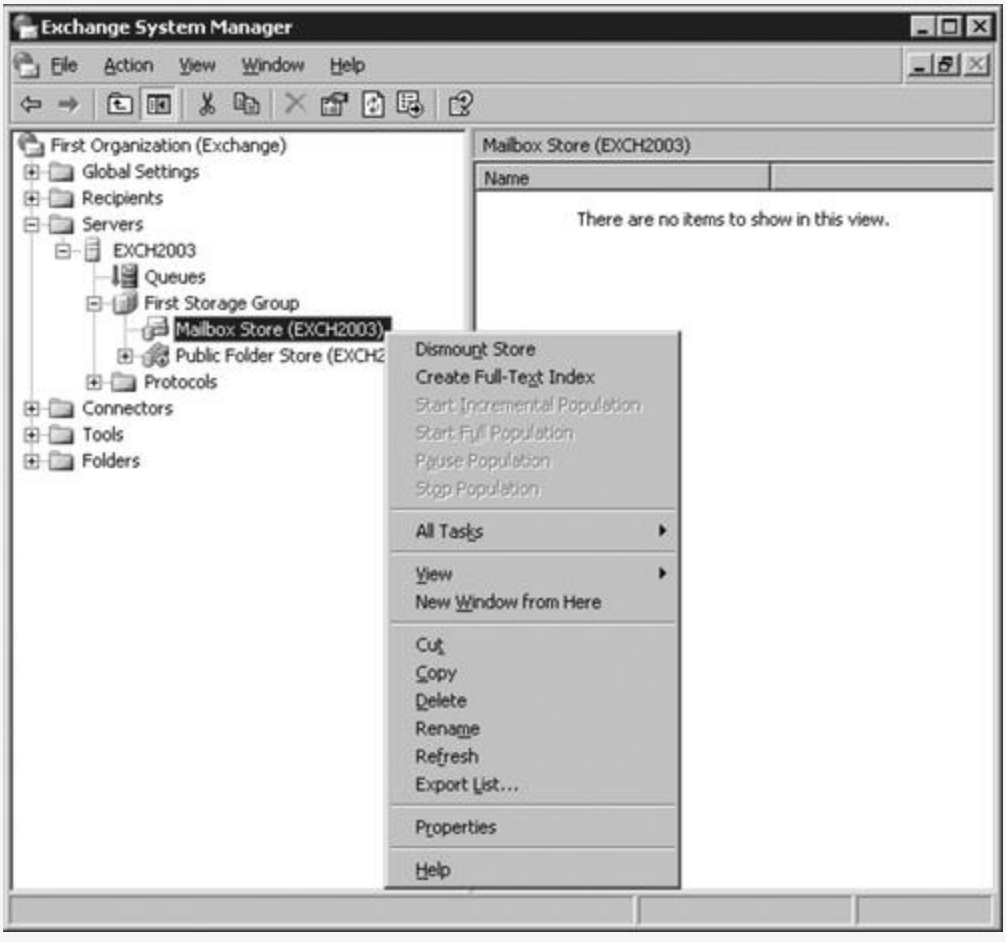


Figure 20-24. Dismounting a store

Right-click on the store again, and select Properties to display the dialog box shown in Figure 20-25. From here,

select the Database tab, and then make sure the “This database can be overwritten by a restore” checkbox is checked.

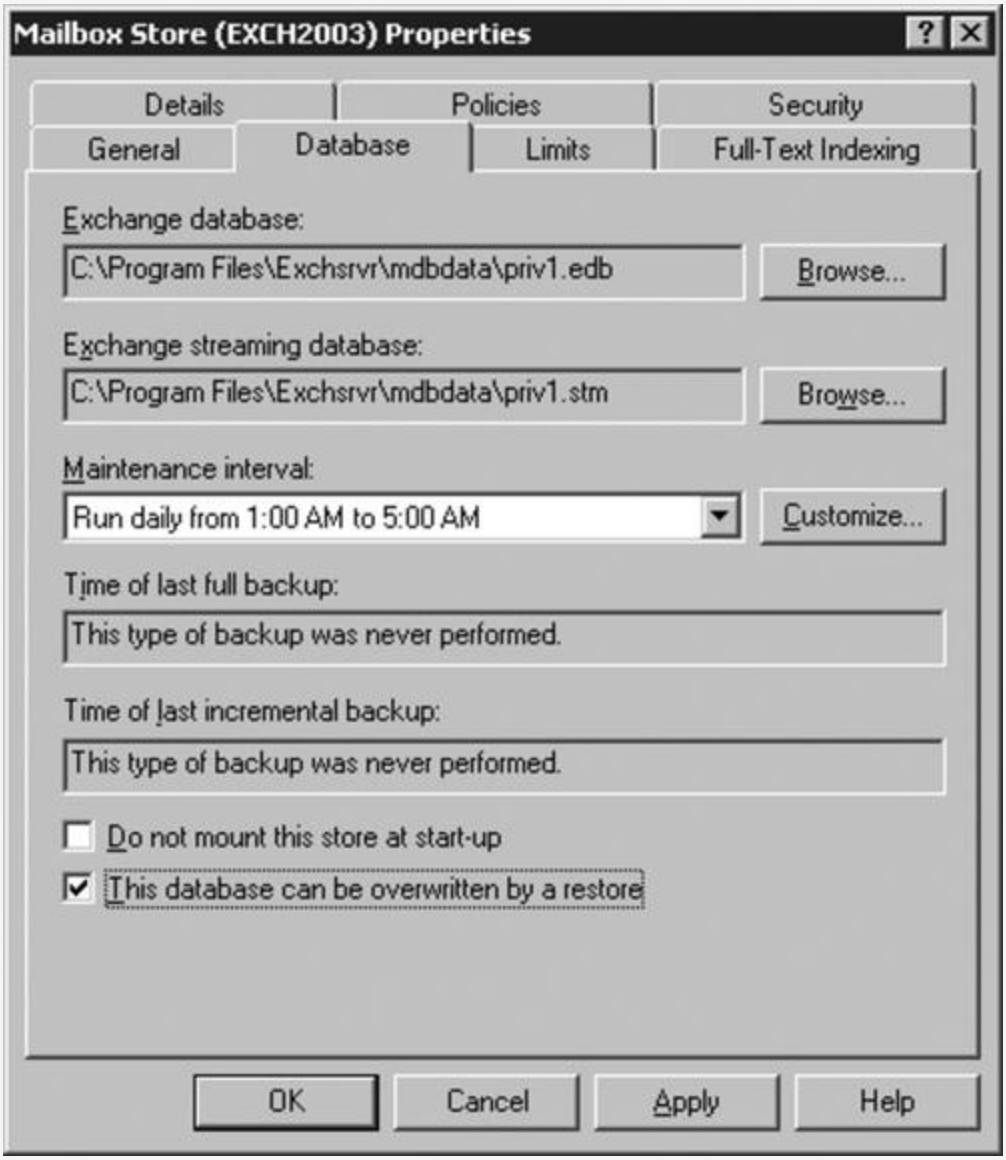


Figure 20-25. Ensuring the database can be overwritten

Next, start `ntbackup` as described in the section “Making a Basic Backup” earlier in this chapter. Select the “Restore and Manage Media” tab. The screen displays the list of your backups, allowing you to select which backup to restore (Figure 20-26). Locate the backup to restore, expand it, and select the items you wish to restore.

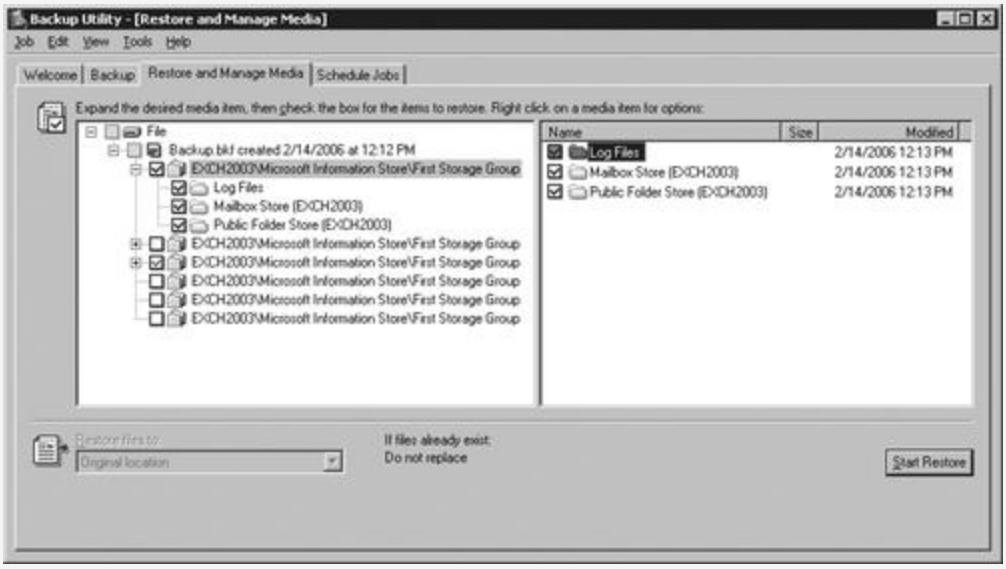


Figure 20-26. Selecting a backup to restore

After you've selected the items to restore, click the Start Restore button to display the dialog in Figure 20-27.

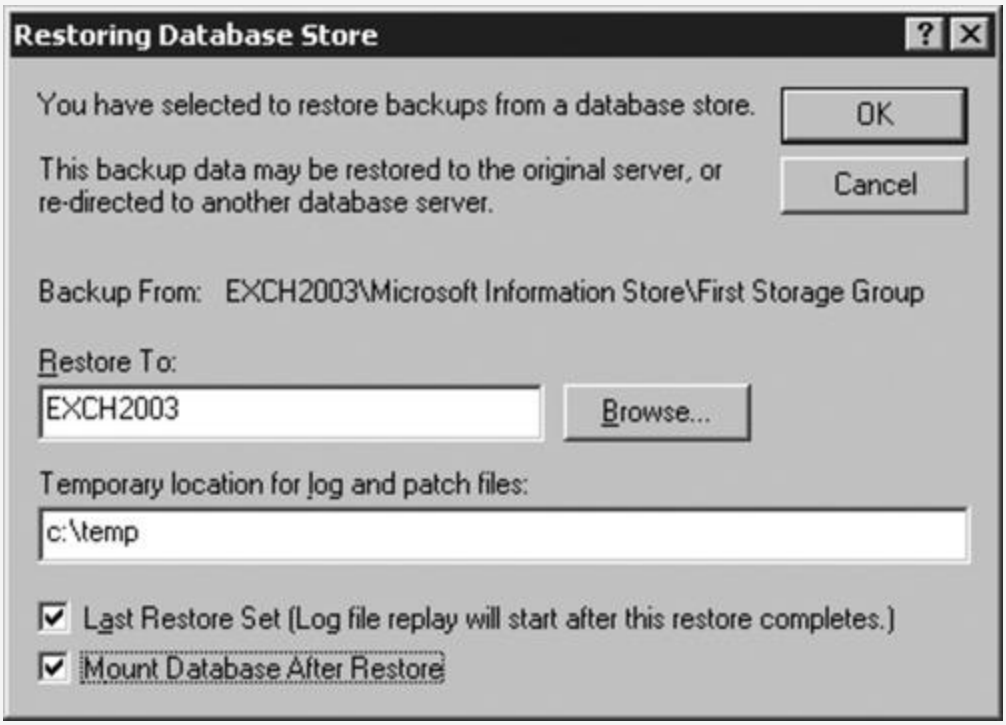


Figure 20-27. Restoring Database Store options

Verify that the proper Exchange Server is in the Restore To location, and then enter a temporary location for the

logfiles. Since this is a basic full restore, and you will use only one backup set, check the Last Restore Set checkbox. If this were part of a normal, differential, or incremental backup and this wasn't the last set in the backup process, you would leave this box unchecked. Don't forget to remount the store. If you select the last checkbox, Mount Databases After Restore, Exchange take cares of this detail for you. Click the OK button when all the options are set and the restore begins. After it is finished, the Restore Progress window shows the status as completed.

If you forget to check the Last Restore Set checkbox, your database is in an inconsistent state. You can manually replay the transaction logs using `eseutil /cc`.

Verify that the restore process was successful by ensuring the Status field is Completed (Figure 20-28). Again, view the report as well as the event logs for additional information.

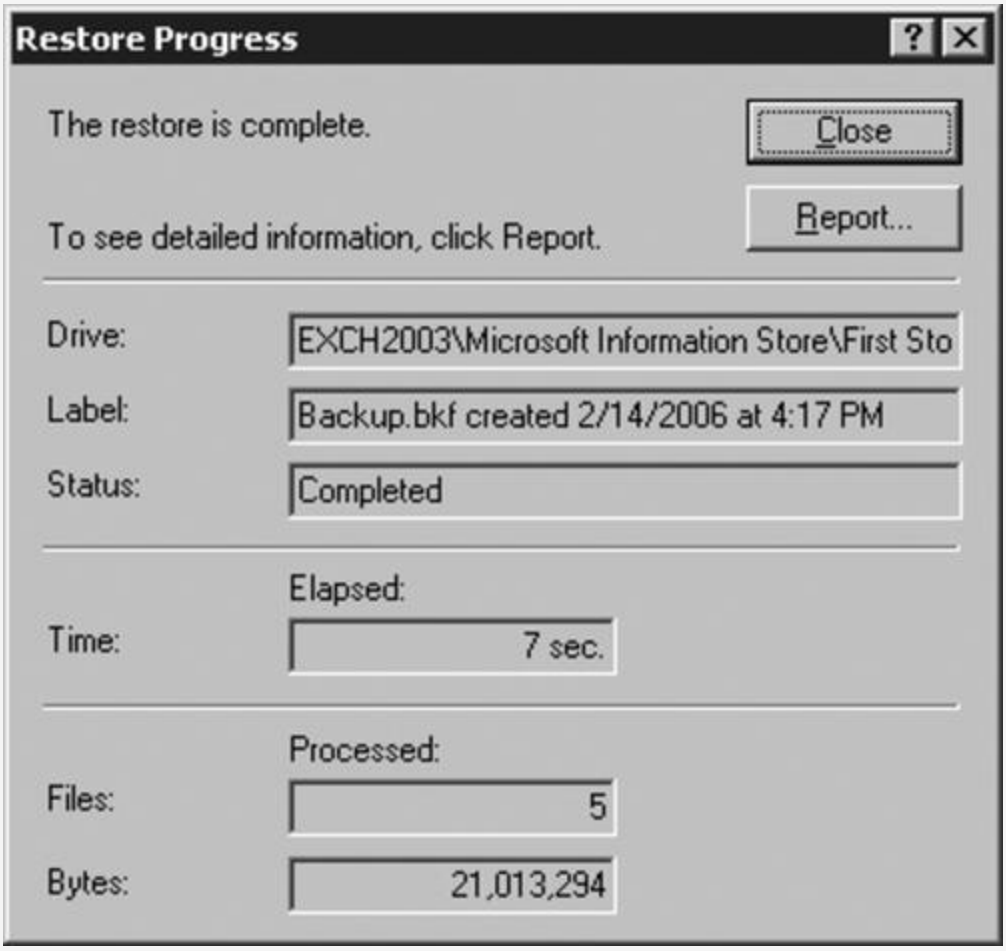


Figure 20-28. Restore progress

Check to make sure the stores have been mounted. Then lastly, try to connect to Exchange using Outlook or Outlook Web Access (OWA) to verify the data you expectto be there is there.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 21. PostgreSQL

As discussed in Chapter 15, any relational database that is to be trusted with production data should be ACID-compliant. PostgreSQL is the most popular open-source relational database that is always ACID-complaint. You can achieve ACID compliance in a MySQL database if you use the InnoDB or the NDB storage engines. (As of this writing, the MySQL team is developing other ACID-compliant storage engines.) With PostgreSQL, all data is stored in an ACID-compliant fashion. PostgreSQL also offers sophisticated features such as point-in-time recovery, tablespaces, checkpoints, hot backups, and write ahead logging for fault tolerance. These are all very good things from a data-protection and data-integrity standpoint.

PostgreSQL Architecture

From a power-user standpoint, PostgreSQL is like any other database. The following terms mean essentially the same in PostgreSQL as they do in any other relational database:

- Database
- Table
- Index
- Row
- Attribute
- Extent
- Partition
- Transaction

Clusters

A PostgreSQL *cluster* is analogous to an instance in other RDBMSs, and each cluster works with one or more databases. It is not typical to have more than one cluster per

database server, but it is possible and is done in some cases. For example, consider a shared hosting environment with different sets of users that should not be given administrative access to each others' data. Since user information is global within a cluster, you would need to create multiple instances to have multiple groups of administrators administering separate databases without giving them the ability to administer another group's database. The `pg_ctl` command is used to start the cluster.

Tablespace

On one hand, a tablespace in PostgreSQL is the same as any other tablespace, as defined in Chapter 15; it is the space into which you put tables and other objects. As in other database systems, a PostgreSQL tablespace also consists of multiple *pagefiles* or *datafiles*. It's how those pagefiles are created that makes PostgreSQL different.

When you create a tablespace in PostgreSQL, you do not specify a list of one or more datafiles; you specify the name of the directory in which you want to store the tablespace. For example, the *default* tablespace is created within the directory where you install the PostgreSQL cluster. Additional tablespaces might reside on an LVM volume or an external RAID volume for performance reasons. When you create the tablespace, PostgreSQL starts creating the first datafile/pagefile. As you create tables in the tablespace and fill them up with data, that pagefile starts getting filled up. Once the pagefile reaches 1 GB in size, PostgreSQL starts creating another pagefile. This continues until the volume where the tablespace resides runs out of space. Additional tablespaces can be created at any time as space is needed, and tables can be moved between tablespaces with a simple `alter table` command.

> **Warning ☠**
>
> Running out of space in a tablespace is a very bad thing: the backend of PostgreSQL crashes, and any open transactions are lost.

Pagefile/Datafile

As mentioned in the tablespace section, PostgreSQL pagefiles/datafiles are similar to datafiles in other databases with one exception. They are not created by the DBA and then assigned to a tablespace. They are created automatically by PostgreSQL as the tablespace needs more room for data.

Startup Scripts

PostgreSQL is started using scripts that are typically custom to the environment. PostgreSQL does not have a global configuration file such as Oracle's *oratab* to keep track of all instances on a machine. Instances on a given machine can be discovered using a process list. On a Unix/Linux machine, `ps –ef` would show something like the following:

```
Need a process list from a PostgreSQL machine
```

System Tables

The system tables, which are stored in the *default* tablespace, can be queried for a list of all databases within the cluster, tablespaces within a database, and so on. This information can then be used for backup purposes. The following command shows a list of databases:

```
postgres=# select datname from pg_database order by datname;
datname
------------------
```

```
calendar
postgres
systemvdm2
template0
template1
 (5 rows)
```

You can also query the system tables for a list of tablespaces with this command:

```
postgres=# select spcname from pg_tablespace order by spcname;
spcname
------------
pg_default
pg_global
(2 rows)
```

Large Objects

PostgreSQL has special datatypes for storing large objects such as image files and large text data. They include ByteA, BLOB and Text. Depending on the configuration, LOB data is stored in special files in the filesystem. If you're running PostgreSQL 8.1 or higher, such data is always backed up when using the `pg_dump` or `pg_dumpall` commands. Previous versions require you to specify the –b option to the `pg_dump` to include this data in a backup.

Rollback Process

PostgreSQL handles the rollback process differently from other RDBMSs. Consider a complex transaction that consists of a single `begin transaction` statement, followed by hundreds of transactional commands and a single `commit` statement. Many RDBMS products would flush the changes to disk as they happen, even if the transaction isn't complete yet. The thought is that 99.9 percent of all transactions are committed anyway, so they may as well be flushed to disk now. The challenge is what happens if someone pulls the plug

on the database: we have an incomplete transaction that is partially written to disk.

This is what the rollback log is for in other RDBMSs. Before changing a block on disk, the before image of that block is stored in the rollback log. If the transaction is committed, its before images are deleted from the rollback log. That means that any blocks in the rollback log are used during crash recovery to return the database to a consistent state.

PostgreSQL handles this differently. Only committed transactions are actually written to disk; uncommitted transactions make changes only to data that is in memory. Another big difference is that changes to tuples in PostgreSQL do not overwrite the previous version of the tuple. PostgreSQL writes a new tuple and marks the old one as inactive. If a transaction completes, its changes are flushed to disk at the next checkpoint, and the inactive tuple is marked for deletion by a vacuum process that happens later. If a transaction *doesn't* complete, it is rolled back by making the new tuple inactive and making the previous version of the tuple active again. This is why PostgreSQL does not need a separate rollback log.

This *vacuum* process happens on a regular basis and deletes from the table all tuples marked for deletion. The vacuum process is extremely important and must be performed. If you ever hit over 4 billion transactions without vacuuming, your database stops working, forcing you to vacuum before you can continue.

> **Warning ☠**
>
> If you don’t occasionally run the vacuum process, your database eventually stops functioning, and you lose any transactions in flight when that happens. Don’t let this happen to you. The good news is that automated vacuuming has been built into PostgreSQL 8.1. It is disabled by default, however.

Write Ahead Log

The write ahead log, or WAL, is very similar to the transaction log in any other database. All transactions are written to the log before they are allowed to be written to disk, allowing the database to redo those transactions in case of crash recovery. The WAL can also be used for point-in-time recovery by replaying transactions that have occurred since the last full backup.

A running PostgreSQL system creates a sequence of WAL records that are divided into WAL segment files, which are normally 16 MB each, although you can specify a different system when building PostgreSQL. These segment files are given numeric names that represent their position in the overall sequence of records. If WAL archiving is not enabled, PostgreSQL creates a few segment files, then recycles them by renaming older segment files with newer segment numbers. This allows enough records to be archived to support hot backups and crash recovery, but does not help if you want to perform point-in-time recovery.

WAL archiving copies the segment files to an alternate location before they are renamed. WAL archiving must be enabled in order to use the WAL for point-in-time recovery. You enable WAL archiving by specifying a value for the `archive_command` parameter on startup. As with other

parameters, this can be specified in *postgresql. conf* or by passing it via an option to the `postmaster` command. If you specify the options in *postgresql.conf*, you should be able to tell `postmaster` to reread the configuration file by issuing a `pg_ctl reload`. If that doesn't work, you need to stop and start the database. The string should be a series of commands designed to copy an individual file to an alternate location. In the string, any `%p` is replaced by the full path of the file to be archived, and any `%f` is replaced by the filename without its path. (Use `%%` if you want to use an actual `%` character in your command.) Here are some example commands from the PostgreSQL manual:

```
$ archive_command = 'cp -i %p /mnt/server/archivedir/%f'
```

This specification would mean that PostgreSQL would copy each WAL segment file to */mnt/server/archivedir* as it is completed. If you want to make sure you're not overwriting any existing files, the following should work on most Unix variants:

```
$ archive_command = 'test ! -f .../%f && cp %p .../%f'
```

It tests to see whether the file is present in the destination before it copies it. Obviously, these examples should be enhanced for use in your own environment.

Alternatively, you can use a custom method for archiving and for other things as well. This example shows such a custom script:

```
$ archive_command = 'mycp.sh %p /mnt/server/archivedir/%f'
```

In addition to copying the files to the appropriate location, the *mycp.sh* script could also do additional checking to suit your requirements. For example:

- Compress or zip the WAL segment file.
- Check that you are not overwriting another segment file.

Backup and Recovery

PostgreSQL offers three methods for backup and recovery: `pg_dump`, `pg_dumpall`, and point-in-time recovery using the `select pg_start_backup` and `select pg_stop_backup` commands. These tools are designed for very different purposes, as the following table shows.

pg_dump	pg_dumpall	pg_start_backup,pg_stop_backup
Supports hot backups	Supports hot backups	Supports hot backups
Backs up a single database, table, or schema	Backs up all databases in a cluster	`pg_start_backup` prepares the entire cluster to be backed up. Actual backup contents are up to your script.
Backups usable by `pg_restore`	Backups not usable by `pg_restore`	Backups not usable by `pg_restore`; they are usable with the point-in-time recovery process
Backups can be in text, `tar`, or `custom` binary format	Backups are in text format	Backups are actual copies of the datafiles and are in whatever format you chose to back up with
Text backups usable by `psql` for restore	All backups usable by `psql` for restore	Not usable with `psql`
Cannot back up global system tables, including users and groups	Backs up global system tables with every backup. Can specify to back up just global tables with the `–g` option.	Backs up the entire cluster
Can only recover database to when you last ran a `pg_dump`	Can only recover database to when you last ran a `pg_dumpall`	Can recover database up to the point of failure if you have all write ahead logs
WAL archiving need not be enabled	WAL archiving need not be enabled	WAL archiving *must* be enabled

The binary format that `pg_dump` can produce is similar to a `dump database` from Sybase and SQL Server. The text format produced by `pg_dump` or `pg_dumpall` is similar to what is produced by `mysqldump` from MySQL. `pg_dump` can dump a single database, where `pg_dumpall` can dump all databases including the system tables. Point-in-time recovery is most like Oracle's `begin/end backup` commands: the commands simply prepare the database to be backed up, rather than creating the backup.

> **Tip ✎**
>
> The reason that you cannot perform a point-in-time recovery with `pg_dump` or `psql` is that these commands actually create a new write ahead log as part of the recovery process. Therefore, they cannot read a write ahead log from a previous iteration of the database.

All methods can be used to safely back up and recover PostgreSQL live. Adding or modifying records/tuples during a backup does not corrupt the backup. Changes to the physical structure of the database (such as adding columns, adding tables, and dropping tables) causes problems during a restore if they are allowed to run during a backup, so these activities are automatically blocked while a database backup is running.

Using pg_dump with pg_restore

`pg_dump` can be used with `pg_restore` or `psql`. This section covers how to use it with `pg_restore`.

Backing up with pg_dump

pg_dump supports two binary formats that are compatible with pg_restore, tar, and custom. The tar format does not allow reordering and/or exclusion of schema elements during restore. The custom archive format allows reordering and/or exclusion of the data load and schema elements, and this format is compressed by default. The default output format of pg_dump is text, which is not compatible with pg_restore; therefore, when running pg_dump, you must specify either the –Ft option for the tar format or the –Fc option for the custom format, if you wish to use the backup with pg_restore.

If you're running a version prior to 8.1, you must specify the –b option to pg_dump and dump in a text format in order to include large objects in the backup. Large objects are included by default in versions 8.1 and later.

Finally, you must specify the name of the database to back up as the last argument to pg_dump. Only pg_dumpall automatically backs up all databases within a cluster.

Putting this together, the following command performs a dump of the *test* database, using the custom archive format (-Fc), backing up BLOB data (-b), and compressing it as much as possible (-Z9):

```
$ pg_dump -Fc -Z9 -b test > db.dmp
```

Restoring with pg_restore

pg_restore is typically used to restore an entire PostgreSQL database in the case of failure, but there are options to restore only selected portions of a database. Let's cover the basic restore first. To restore a database using the dump created with the previous example, issue the following command. It tells pg_restore to restore the *test* database from the *db.dmp* dump file. Compression and backup format are automatically detected, and BLOB data

is included if it was in the backup. This command assumes that the test database is already created and ready to be restored:

```
$ pg_restore-d test db.dmp
```

If the database that you wish to restore has not been created, you can ask `pg_restore` to create it for you:

```
$ pg_restore-C –d test db.dmp
```

You also might want to restore just one table or trigger. In this case, you can specify that table or trigger via an option on the command line. This command restores just the special table of the *test* database:

```
$ pg_restore test –t special db.dmp
```

This command restores just the `specialtrigger` trigger of the *test* database:

```
$ pg_restore test –T specialtrigger db.dmp
```

If you want to restore several database objects, but not all of them, you can create a list of everything in the dump, edit the list, then tell `pg_restore` to restore just what's in the list. First, create a list of what's in the dump using the following command:

```
$ pg_restore -l db.dump > dump.list
```

This produces a file that looks something like this:

```
;
; Archive created at Sun Jul 02 22:28:36 2006
;     dbname: test
;     TOC Entries: 50
;     Compression: 9
;     Dump Version: 1.4-0
;     Format: CUSTOM
;
;
; Selected TOC Entries:
;
2; 145344 TABLE species postgres
```

```
3; 145344 ACL species
4; 145359 TABLE nt_header postgres
5; 145359 ACL nt_header
6; 145402 TABLE species_records postgres
7; 145402 ACL species_records
8; 145416 TABLE ss_old postgres
9; 145416 ACL ss_old
10; 145433 TABLE map_resolutions postgres
11; 145433 ACL map_resolutions
12; 145443 TABLE hs_old postgres
13; 145443 ACL hs_old
```

You can comment out lines by preceding them with a semicolon, then tell the `pg_restore` command to restore the tables that it finds in the dump listing. You can also reorder objects by changing the order in which they appear in the output.

```
$ pg_restore -L dump.list db.dump
```

Using pg_dump with psql

The default backup format for `pg_dump` is text. Therefore, if you issue a `pg_dump` command without specifying a backup format, you create a text file that contains the commands necessary to rebuild the database you backed up. You can also, of course, specify the text output format on the command line.

```
$ pg_dump -Fp -b test > db.sql
```

This command creates a text file containing all the commands necessary to rebuild the database. Because the –b option is specified, the text also contains BLOB data converted to text. (This is not necessary in versions 8.1 and higher.) The text format does not support compression; however, you can compress the data yourself by running it through `gzip` or a similar command:

```
$ pg_dump -Fp -b test | gzip –c > db.sql.gz
```

To restore from this backup, pass it to the `psql` command:

```
$ psql -d test -f db.sql
```

If you compressed it, you need to uncompress it before passing it to `psql`:

```
$ gzip –dc db.sql.gz | psql -d test
```

Obviously, you can edit the SQL file, deleting any tables or other objects that you don’t want to restore.

Using pg_dumpall with psql

`pg_dump` backs up all databases within a cluster, and the *only* backup format for `pg_dumpall` is also text. The most common use of `pg_dumpall` is to create a single database backup of every database in a cluster, using the following command:

```
$ pg_dumpall > all-databases.sql
```

This command creates a text file containing all the commands necessary to rebuild every database in the cluster, including large object data. The text format does not support compression but you can compress the data yourself by running it through `gzip` or a similar command:

```
$ pg_dumpall |gzip –c > all-databases.sql.gz
```

If you’re using `pg_dump` or the point-in-time recovery method to back up your databases, you can use `pg_dumpall` to back up just the global objects, such as users and groups. This is done by specifying the –g option to the `pg_dumpall` command:

```
$ pg_dumpall –g > global-objects.sql
```

You can also compress this backup as well:

```
$ pg_dumpall –g | gzip –c > global-objects.sql.gz
```

This command creates a text file containing all the commands necessary to rebuild just the system tables for a

cluster.

To restore from either of these backups, pass the backup file to the `psql` command:

```
$ psql -f all-databases.sql
$psql -f global-objects.sql
```

If you compressed these backups, you need to uncompress them before passing them to `psql`:

```
$ gzip –dc all-databases.sql.gz | psql
$gzip –dc global-objects.sql.gz |psql
```

Obviously, you can edit the SQL file, deleting any tables or other objects that you don’t want to restore.

Point-in-Time Recovery

Point-in-time, or PIT, recovery in PostgreSQL is very different from either `pg_dump` or `pg_dumpall`. However, if you're familiar with how to do hot backups in Oracle without using `rman`, this is very similar. It's also not that complicated.

Creating a Backup to Use with Point-in-Time Recovery

If you're going to do PIT recovery, you have to first enable WAL archiving. If you haven't done that, review the write ahead log section to see how to do so.

Once you've enabled WAL archiving, the process is simple:

1. Tell PostgreSQL that you're going to start an external backup (`select pg_start_backup`).
2. Perform the actual backup.
3. Tell PostgreSQL that you're finished with the external backup (`select pg_stop_backup`).

Let's look at that in more detail. The first thing you have to do is to tell PostgreSQL that you're going to start a hot backup. Connect to any database in the cluster as superuser, and issue the following SQL command:

```
> select pg_start_backup('label');
```

For *label*, use any string that you want to identify the backup, such as the full path of where you're going to put the backup. PostgreSQL then puts a file called *backup_label* in the cluster directory. This file contains information about the backup such as start/stop time of the backup and the WAL segments written while backup was being done. You have now told PostgreSQL that you plan to perform an external backup. You

can then copy the files underneath the cluster directory any way you wish, while probably excluding the *pg_xlog* directory to save some space. Methods can include everything from creating a `tar` archive on disk to running a commercial backup utility to back it up to tape or disk.

Once the backup is done, you need to tell PostgreSQL that you're done. Connect to any database in the cluster as superuser, and issue this SQL command:

```
> select pg_stop_backup;
```

The write ahead log is used during recovery to correct any inconsistencies created by backing up the datafiles/pagefiles while they were being used. It is therefore essential that you have every WAL segment that was active from the time you selected `pg_start_backup` and `pg_stop_backup`. If you don't have every one of these WAL segments, you cannot use that backup.

The final WAL segment is archived as soon as you generate enough changes to fill it. However, this could be several hours or several days, depending on the level of change within your database. If you lost your entire database directory (including the *pg_xlog* subdirectory), you couldn't use the backup you just created because the last WAL segment it needs would be lost. It would be nice if you could force PostgreSQL to switch WAL segments, but this is not available as of this writing. If you could do that, the current WAL segment would be archived immediately, and you would be sure that you could use this current backup.

Suppose that you do find yourself in a recovery scenario in which you cannot use the most recent backup because you lost the disk drive where the database and the *pg_xlog* directory resided. What would you do? The option that results in the least amount of data loss would be to recover from the previous backup (which you hopefully kept) and use the archived transaction logs to roll forward to the latest log you have available.

If you'd like to see which WAL segments are needed for the hot backup, you can examine the *.backup* file that is created in the WAL archive directory. The filename would look something like this:

000000010000001A00000012.00535CD8.backup

In this example, *000000010000001A00000012* was the current WAL log when the backup was started (`pg_start_backup`), and *00535CD8* is the checkpoint record. If you examine this file, you'll see something like the following:

```
START WAL LOCATION: 1A/12535CD8 (file 000000010000001A00000012)
STOP WAL LOCATION: 1A/131C407C (file 000000010000001A00000013)
CHECKPOINT LOCATION: 1A/12535CD8
START TIME: 2006-07-11 11:59:02 BST
LABEL: DBBACK-FILE-20060614-132.tar.gz
STOP TIME: 2006-07-11 12:04:21 BST
```

> **Warning ☠**
>
> To be able to restore with this backup, you absolutely must have the file specified in `START WAL LOCATION` (e.g., *000000010000001A00000012*), the file specified in `STOP WAL LOCATION` (e.g., *00000001000000-1A00000013*), and any files in between. (In this example, there are no files in between the start and stop WAL segments.)

Once you've done all this, your backup is done.

Restoring from a Point-in-Time Backup

The steps to restoring from a PIT backup are relatively simple. Yes, they're more complicated than a `pg_restore` or `psql` command, but there are two advantages to restoring this way. The first is that you can perform a multithreaded restore, greatly increasing the speed of a large restore. The second advantage is that you can recover right up to the

point of failure. If those advantages aren't important to you, then use one of the other methods. Here's the procedure:

1. Stop the `postmaster` if it's running.
2. If you've got the space to do so, and if it's still available, you might want to back up the whole cluster data directory (and any other place you put tablespaces) to some alternate location. At a minimum, you should copy the contents of the *pg_xlog* subdirectory because it may contain unarchived WAL files.
3. Clean out the cluster data directory and any other directories into which you will be copying data, including *pg_xlog*.
4. Restore the database files from the backup taken earlier, with the appropriate ownership and permissions. Make sure that you don't overwrite any symbolic links that you created.
5. If your restore copied any files into *pg_xlog*, remove them; these are from the backed up database and are now obsolete. If you didn't back up *pg_xlog*, be sure to recreate it and the *pg_xlog/archive_status* subdirectory as well.
6. Copy (not move) back into *pg_xlog* any unarchived WAL files that you saved in step 2, or that you saved immediately after the backup.
7. Create a file *recovery.conf* in the cluster data directory. This file contains a `restore_command` that is essentially the opposite of the `archive_command` that you used for archiving WAL segment files. For example, suppose you use this `archive_command` during normal operations:
8. ```
   % archive_command = 'cp -i %p /mnt/server/archivedir/%f'
   ```
9. The reverse of that would be the `restore_command`:
10. ```
    % restore_command = 'cp /mnt/server/archivedir/%f %p'
    ```
11. Put the appropriate command into the *recovery.conf* file.

12. It might also be a good idea to modify *pg_hba.conf* so that ordinary users can't connect as soon as the recovery is complete. Once you've verified that the restore worked, you can undo this change and allow users access to the database.
13. Start the postmaster. It automatically goes into recovery mode and processes the appropriate WAL files. Once it's done, it renames *recovery.conf* to *recovery.done* and then commences normal database operations.
14. Check the database to make sure all is well. If not, return to step 1. If so, return everything to normal. If you denied users access in step 8, allow them to connect by restoring *pg_hba.conf* to its normal status and doing a `pg_ctl reload`.

Please note that this method is an all-or-nothing method. You cannot use the PIT method to restore an individual table, tablespace, or database. You also cannot restore to a point in time before the backup was taken.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 22. MySQL

MySQL is a very popular open-source database with a large user community behind it. From a user perspective, interacting with MySQL is similar to interacting with any other RDBMS. From a DBA perspective, MySQL is definitely a horse of a different color.

The big difference between MySQL and other databases is its concept of *storage engines*, where a storage engine does the job of actually storing the data on disk. MySQL has a *pluggable storage engine* architecture. That is, it can support any storage engine that interacts with its API. As of this writing, MySQL has more than 10 supported storage engines in, including MyISAM, InnoDB, BDB, Memory, Merge, Archive, Federated, NDB, CSV, BlackHole, and Example. There are also a few storage engines in the works as of this writing, including PBXT, SolidDB, and one code-named Falcon.

> **Tip ✎**
>
> MySQL's pluggable storage engine architecture also makes writing a chapter on backup and recovery of MySQL somewhat difficult. Many major storage engines have their own backup and recovery method, creating many options and "if then" statements. I strongly encourage you to read the documentation for the storage engine you are using to ensure that you are very familiar with backup and recovery of that storage engine. Also, when you're just starting out, make sure you join the mailing list and ask for any usual first-timer mistakes that you can avoid.

The statements in this chapter have been tested only with MySQL versions 5.0 and 5.1. They may or may not work on previous or later versions. I also cover only the MyISAM,

InnoDB, and NDB storage engines in this chapter. MyISAM is the default storage engine, InnoDB has the most ACID-compliant features as of this writing, and NDB (also ACID-compliant) is the memory-resident storage engine used with MySQL cluster. (ACID compliance is covered in more detail in Chapter 15.)

MySQL Architecture

MySQL architecture, especially the storage and backup part, is driven by the storage engine that you choose, but all storage engines share certain architectural elements.

Shared Architectural Elements

The architectural elements in the following sections are the same regardless of which storage engine you choose.

The power user's view

From the power user's standpoint, MySQL is like any other database. The following terms mean the same in MySQL as they do in any other relational database:

- Database
- Table
- Index
- Row
- Attribute
- Partition

Instances

A MySQL instance is the same as any other database instance. It controls access to one or more MySQL databases. A MySQL instance is available after the `mysqld` process is started. Multiple instances of MySQL are possible as long as

each has separate data directories and listens on different TCP ports and sockets. The MySQL instance manager can be used to monitor and manage these instances.

Startup scripts

MySQL can be started via any script, but MySQL does provide a few scripts for you. The included *mysqld_safe* script has historically been the recommended way to start `mysqld`. As the name of the script implies, it adds some safety features such as restarting the server when an error occurs and logging runtime information to an error logfile. There is also now the *mysql.server* script that's automatically installed in the appropriate place in Linux systems and the Instance Manager that can be used to manage multiple instances. (The Instance Manager is scheduled to replace *mysqld_safe* at some point.)

Also important is the *my.cnf* file, which is typically stored in */etc* on Unix/Linux systems. It is an optional, although often used, file that stores a number of startup options for MySQL databases.

Databases and tablespaces

The `show databases` command can be used to obtain a list of all databases within the instance. This information can then be used for backup purposes.

> **Tip ✎**
>
> This command gets its information from a directory listing, not a query of a table in the *mysql* database.

If you're using the InnoDB storage engine, you can also query InnoDB for a list of tablespaces:

```
> show variables like 'innodb_data_file_path%'
+-----------------------+------------------------+
```

```
| Variable_name          | Value                    |
+------------------------+--------------------------+
| innodb_data_file_path | ibdata1:10M:autoextend |
+------------------------+--------------------------+
```

Large objects

MySQL has special datatypes for storing large objects such as image files and large text data. The four datatypes for storing binary large objects, or BLOBs, are *tinyblob*, *blob*, *mediumblob*, and *longblob*, differing only in the maximum length of the values they can hold. The four datatypes for storing large text objects are *tinytext*, *text*, *mediumtext*, and *longtext*. All large object data is stored inside the database, so it is included in any database dumps.

Binary log

The *binary log* contains a history of SQL statements that changed data, and its primary purposes are point-in-time recovery and replication. The binary log can be applied against a consistent backup of a MySQL database to provide up-to-the-minute recovery of a database, and it is also used to send changes from a replication master to its slave(s).

The binary log contains only SQL statements, and SQL statements can be applied only against a consistent database. The binary log therefore cannot be used to bring the database back to a consistent state after a crash. It also cannot be used to create a consistent backup from an inconsistent backup of a running database, the way Oracle's redo logs can be used with the `begin backup` and `end backup` commands. This is why MySQL administrators do not refer to the binary log as a transaction log; they see crash recovery as the primary purpose of a transaction log.

The binary log is started by default in some systems and not in others. To use the binary log, you should specify a value for the `log-bin=` *`base_name`* option in the *my.cnf* file:

```
[mysqld]
log-bin[=base_name]
```

> **Tip ✎**
>
> It’s considered best practice to specify a full pathname for *base_name* (for example, */backups/binarylog*) so that you can send the logs to a directory owned by the user running `mysqld`. In this example, */backups* should be writeable by the user running `mysqld`.

MySQL takes the base name you supplied, appends a number to it (for example, *.001*), and makes a series of binary logs with unique numeric extensions. For example, if you specify `/backups/binarylog` as *base_name*, it creates */backups/binarylog.001*, then */ backups/binarylog.002*, and so on. You should keep these logs for point-in-time recovery. Make sure that they are backed up with your normal filesystem backups.

Make sure to investigate the `max_binlog_cache_size` variable, which is set to 4 GB by default. If this size is not large enough to contain all the statements from a given transaction, that transaction fails and is rolled back.

MyISAM Storage Engine

The MyISAM storage engine is the default storage engine in MySQL, and it is the required storage engine for system tables. The MyISAM storage engine, while very fast, is not ACID-compliant (atomicity, consistency, independence, durability). It has no concept of transactions, so only changes made in a single SQL statement are atomic. It does not support foreign keys and other features necessary to ensure referential integrity, so it doesn’t pass the *consistency* test. It supports only table-level locking, so *independence* is also hard to achieve. Finally, if the server were to crash in the middle of a number of changes, it’s highly possible that the database

would be left in an inconsistent state, requiring a full recovery to continue—making it not very *durable*. A good backup and the use of the binary log would allow you to recover up to the point of failure, but this would take a lot longer than the crash recovery process of an ACID-compliant database.

Each MyISAM table is stored in a series of three files. A file with a *.FRM* extension stores the table format. Data is stored in a file with an extension of *.MYD* (MYData), and the index file has a *.MYI* (MYIndex) extension. (You may find these filenames in upper- or lowercase.) Since tables cannot be distributed across multiple *.MYD* files, each table is limited to the maximum size of a file in the operating system. This used to be 2 GB, but that limitation has been removed from every major operating system.

MyISAM files are stored underneath the `datadir`. Each database creates a subdirectory underneath `datadir` and stores its files there. `datadir` can be determined using the following command:

```
> show variables like 'datadir';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| datadir       | /var/lib/mysql/ |
+---------------+-----------------+
```

If you wish to perform a backup of a MyISAM table, you should really obtain a read lock on it before doing so. This lock prevents any update, insert, or delete statements during the backup.

InnoDB Storage Engine

If your application calls for ACID-compliance, InnoDB is the storage engine for you. It is fully ACID-compliant, with full support for transactions, tablespaces, rollbacks, foreign keys, and hot backups. It is not as fast as the MyISAM storage

engine, but for good reason. It takes time to perform all the checks necessary for ACID-compliance.

If you're currently using MyISAM tables and you wish to migrate them to InnoDB for integrity purposes, it's a relatively simple process. You can use `alter table ... engine=innodb`, or create an empty InnoDB table with identical definitions and insert the rows with `insert into ... select * from ...`. Please consult the MySQL documentation for ways to speed up large tables importation.

> **Tip ✎**
>
> For data integrity reasons, it is important on Linux to disable the write cache. The command `hdparm -w0 /dev/hda` should work on most ATAPI disks.

If you'd like to use InnoDB for all nonsystem tables, simply add the line `default-storage-engine=innodb` to the `[mysqld]` section of your server option file.

> **Warning ☠**
>
> Do not convert *mysql* system tables to the InnoDB storage engine. The system tables must always use the MyISAM storage engine.

Like MyISAM tables, InnoDB files are typically stored underneath the `datadir`. Each database creates a subdirectory underneath `datadir` and stores its files there. A default tablespace called `ibdata1` is stored directly in `datadir`. If you are performing file-level backups, that file must be backed up as well. `datadir` can be determined using the following command:

```
> show variables like 'datadir';
+---------------+-----------------+
| Variable_name | Value           |
```

```
+--------------+-----------------+
| datadir      | /var/lib/mysql/ |
+--------------+-----------------+
```

> **Warning ☠**
>
> Do not forget to back up the default tablespace file *<datadir>/ibdata1*. It is very common for people to back up the files in the database directories (e.g., *<datadir>/<database>*) and forget the default tablespace file in *<datadir>*. Also, do not assume that all datafiles are stored underneath `datadir` .

Concerns about InnoDB

InnoDB is a commercial product that was being licensed from Innobase, a Finland-based company. Oracle purchased Innobase in 2005, causing concerns that Oracle would stop development of the InnoDB storage engine. Those fears appear unfounded, especially if you read Oracle's official statement about the matter (http://www.oracle.com/innodb/index.html):

"Oracle has long been a supporter of open source software such as Linux and Apache," said Charles Rozwat, Oracle's Executive Vice President in charge of Database and Middleware Technology. "Innobase is an innovative small company that develops open source database technology. Oracle intends to continue developing the InnoDB technology and expand our commitment to open-source software. Oracle has already developed and contributed an open source clustered file system to Linux. We expect to make additional contributions in the future."

Transactions

A *transaction* in a MySQL database that's using the InnoDB storage engine is essentially the same as a transaction in any other database, except that InnoDB does not support all the same isolation levels that other databases do. You can create simple and complex transactions, and all updates are logged to the InnoDB transaction log and/or the rollback segment. The SQL statements are also recorded to the MySQL binary log. The transaction log and rollback segment are used for crash recovery of the database, and the binary log allows you to reply SQL statements against a consistent database backup to redo those SQL statements.

Tablespace

A *tablespace* in the InnoDB or NDB storage engines is similar to an Oracle tablespace; it consists of one or more datafiles. NDB tablespaces can be added via a `create tablespace` or `alter tablespace` command. (These commands were added in MySQL 5.1.) Currently, InnoDB tablespaces are added by editing the *my.cnf* file.

Datafile

A *datafile* is a part of an InnoDB tablespace. It is added to the tablespace via a `create tablespace` or an `alter tablespace` command.

Rollback segment or log group

The InnoDB *rollback segment* behaves similarly to the rollback segment in Oracle. It uses this information to perform undo operations that are needed in the rollback of a transaction. It also is used to build earlier versions of a row for a consistent read. Information is kept in the rollback segment until it is no longer needed for rollback. A *log group* performs this function in the NDB storage engine.

Transaction log

InnoDB has its own transaction log that records transactions for InnoDB tables, and is used with the rollback segment to ensure database consistency. However, it is not used to replay transactions after a consistent database backup, so InnoDB transaction logs are not archived. Therefore, `innodb_log_archive` is set to `0`, which disables log archiving in InnoDB.

Other Storage Engines

There are a number of other storage engines available in MySQL. Some are available as of this writing, and others are in development.

NDB

: NDB is a storage engine that typically resides entirely in RAM and is used by MySQL cluster. MySQL 5.1 introduced the ability to store nonindexed tables on columns using a log group for undo and a tablespace for storage. Other objects reside only in RAM and are written to disk during a checkpoint and during shutdown. The NDB storage engine also supports hot backups, which are covered later in the section “MySQL Cluster Hot Backup and Recovery.” NDB cluster supports only the `read committed` transaction isolation level.

PBXT

: PBXT is an independently developed storage engine. It will be available as a plug-in for MySQL at some point. You can read about it at http://forge.mysql.com/projects/view.php?id=43.

SolidDB

SolidDB is another pluggable storage engine, developed by Solid Information Technology. You can read about it at http://forge.mysql.com/projects/view.php?id=139.

Falcon

Falcon is the code name for a new transactional storage engine from MySQL. It is developed by Jim Starkey. You can read about it at http://forge.mysql.com/wiki/Falcon.

MySQL Backup and Recovery Methodologies

There are two methodologies you can use to back up MySQL databases without spending money. The first is to create a series of SQL statements that rebuild one or more databases. This is usually accomplished via the `mysqldump` script. The second is to copy the database files themselves if you can ensure they're not being changed during the backup. This is currently possible only for MyISAM archive tables using the `mysqlhotcopy` script.

> **Tip ✎**
>
> Whichever backup method you choose, you should ensure that you have enabled the binary log prior to beginning backups. This allows you to perform a point-in-time recovery if you choose to do so.

If you're using InnoDB tables, you can also purchase a commercial hot backup tool from Oracle. This tool is not covered in this book, but it offers a number of advantages that help warrant its price.

SQL-Level Backup and Recovery

`mysqldump` is designed to automate the creation of a SQL-level backup of one or more databases in a MySQL environment. It produces a human-readable text file containing all the SQL commands needed to recover MySQL.

To do a SQL-level dump, you should lock all tables before you start and unlock them when you're done. This can be done using SQL commands.

If you're backing up MyISAM tables, you can use the `--lock-tables` or `--lock-all-tables` options to `mysqldump`. The `--lock-tables` option locks tables one at a time and so does not guarantee table consistency across multiple tables. The `--lock-all-tables` option locks all tables in all databases during the time of the backup, ensuring consistency across multiple databases; however, this may have a greater impact on your applications because the lock remains in effect for a much longer time. Although the databases stay up while backing up with these options, the locking of tables can have a dramatic impact on the usability of the database. For example, inserts, updates, and deletes are not allowed to execute when a table is locked. They lock, or freeze, and then execute once the backup is completed.

If you're backing up InnoDB tables, you should use the `--single-transaction` option instead. It creates a short lock in the beginning of the dump but maintains consistency through the rest of the backup without having to lock any tables.

> **Tip ✎**
>
> The `--lock-tables`, `--lock-all-tables`, and `--single- transaction` options are mutually exclusive. Specifying more than one of these options may cause your dump to fail. It may also cause one or more of the specified options to be ignored.

If you are using the binary log to restore your database up to the point of failure, you'll also want to add `--flush-logs` and `--master-data=2`. Flushing the logs when you make your full backup starts a new binary logfile, ensuring that the new binary log starts with SQL statements that were issued since the backup. The `--master-data=2` option

ensures the dump file contains the name of the new current binary log.

Backing up MyISAM tables

The following command creates a SQL dump of all MySQL databases in a given instance, locks all tables as necessary, flushes the logs, and writes the name of the current binary log to the output. The second command causes MySQL to switch binary logfiles so that you can back up the logfile that was used during the backup.

```
$ mysqldump --all-databases --lock-all-tables --flush-logs --master-data=2
> all_databases.sql
$mysqladmin --flush-logs
```

Backing up InnoDB tables

If you are backing up InnoDB tables, you should replace the `--lock-all-tables` option with `-- single-transaction`. This treats the `mysqldump` process as a transaction, complete with `begin transaction` and `end transaction` statements. This first command backs up all databases, flushes the logs, and write the name of the current binary log to the output. The second command backs up just the `mysql` database using the `--lock-all-tables` option, which is the proper way to back up that database because it uses the MyISAM storage engine. The third command causes MySQL to switch binary logfiles so that you can back up the logfile that was used during the backup.

```
$ mysqldump --all-databases --single-transaction --flush-logs --master-data=2
> all_databases.sql
$ mysqldump --database=mysql --lock-all-tables --flush-logs --master-data=2
> system.sql
$mysqladmin --flush-logs
```

Repairing corrupted MyISAM tables

Databases are restored for two reasons. The first reason is physical damage; the database was deleted, or the device on which it was being stored was damaged. The second reason is logical damage; one or more tables in the database got corrupted. If you have corrupted MyISAM tables, you should try to repair the tables before restoring them. It should be faster. The following command fixes most problems. First, `cd` to the directory that holds the MyISAM datafiles, then run this:

```
$ myisamchk -r -qtable_name
```

If that doesn't work, try this command; it fixes some problems that the quick repair just shown does not:

```
$ myisamchk --safe-recovertable_name
```

SQL-level MySQL restores

Restoring MySQL databases that were backed up with `mysqldump` is easy. Simply tell the `mysql` command to read the file:

```
$ mysql –u username –p < all_databases.sql
```

Of course, you can edit the output of `mysqldump`, deleting any databases or tables that you don't want to restore.

If you want to apply the changes made to the database since the time of the backup, follow the instructions in the section "Using Point-in-Time Recovery" later in this chapter.

File-Level Backup and Recovery

Since MyISAM tables are ultimately stored in files on the filesystem, you can copy them using a backup program if you can ensure that they're not being changed.

Build your own file-level backup

There are dozens of ways that you can use this method to back up your MySQL databases, but they're all essentially the same: get MySQL to stand still for a moment while you copy its datafiles. This can be done in three different ways:

- Flush tables and logs, get a read lock on the tables, back up the datafiles, then unlock the tables.
- Flush tables and logs, get a read lock on the tables, make a snapshot, then unlock the tables.
- Shut down the database, back up the datafiles (or take a snapshot), then start the database.

If you're going to back up MyISAM files live, you need to make sure MySQL isn't changing them. To do this, make sure the database has been flushed to disk and that you've obtained a read lock on the appropriate tables. Then you can do whatever you want to back up the files.

To obtain a read lock and flush MySQL to disk, issue the following commands:

```
mysql> flush tables with read lock;
mysql>flush logs
```

Once you've done that, you can do whatever you want to create the backup, realizing that any inserts, updates, or deletes will be blocked during this operation. (Therefore, you might want to do the backup as quickly as possible, depending on your environment.) Suggestions offered by the MySQL community include a typical backup or the creation of a snapshot.

You can do the typical backup just about any way you want to. You can create a `tar` or `cpio` backup image on disk or tape. You can also just run your typical open-source or commercial backup automation program.

Make sure to unlock the tables when you're done:

```
mysql> unlock tables;
```

Another interesting idea is to use the Linux Logical Volume Manger (LVM) to quickly create a snapshot. This allows you to take a quick virtual backup, then take as long as you want to back up that snapshot to other media. You still need to use `mysql` to lock the tables and flush the logs:

```
mysql> flush tables with read lock;
mysql>flush logs
```

Next, tell LVM to create the snapshot, where *`somename`* is a string you assign to the snapshot, *`somesize`* is how large you allow the cache to get for this snapshot, and *`volgroup/lvol`* is the name of the volume group and logical volume, such as `vg01/lvol1`:

```
# lvcreate --snapshot --size=somesize --name=somename/dev/volgroup/lvol
```

Once you've created the snapshot, mount it:

```
# mkdir -p /mnt/somename
# mount -o -ro /dev/volgroup/somename/mnt/somename
```

Once you've done that, you can take all the time you want to back up `/mnt/` *`somename`*, unmounting it when you're done. Then remove the snapshot with the following command:

```
# lvremove -f /dev/volgroup/somename
```

Finally, make sure you unlock the tables when you have finished. The following command does that:

```
mysql> unlock tables;
```

You could, of course, also perform a *cold backup*. That is, you could back up the database while it is down. The only preparation for a cold backup is to stop `mysqld`. Once your backup is done, or a snapshot is taken, you can start up the database again.

File-level backup with mysqlhotcopy

If you're using MyISAM or Archive tables, the `mysqlhotcopy` command automates the creation of a file-level backup for you. It locks the database and copies the database files to another location. To execute a `mysqlhotcopy` backup, run the following command, where *`database_name`* is the database you wish to back up, and *`somedirectory`* is the name of the directory you want to back up to:

```
$ mysqlhotcopy database_name/somedirectory
```

One challenge with this method is that the default authentication method requires you to pass the username and password on the command line, which is not very secure. A workaround for this is to edit the `[client]` subheader in the *my.cnf* file to include the following lines:

```
[client]
username=root
password=yourpassword
```

Restoring from file-level backups

Restoring from file-level backups is easy. All you have to do is stop `mysqld` and follow your typical restore procedure to restore the files from the backup system into your *mysql* directory. One downside to this method is that there's no easy way to merge data from such backups into existing tables. You can restore them to a different directory and run `mysqldump` against them to create SQL statements to merge them.

If you want to apply the changes made to the database since the time of the backup, follow the instructions in the next section.

MySQL Projects

The MySQL community is working on a few things for the future. One is an online backup tool designed to make `mysqldump` and `mysqlhotcopy` obsolete. It is at least a year away from being generally available as of this writing (hopefully, it will be available in 2007), but you can read about it at http://forge.mysql.com/wiki/OnlineBackup. Another project is another ACID-compliant storage engine code-named *Falcon*, which should also be generally available sometime in 2007.

Using Point-in-Time Recovery

If you enabled the binary log, you have the option of a point-in-time recovery, regardless of which backup and recovery method you choose. If the binary log is running, you will find a series of files using the string that you specified in the `--log-bin=` *`base_name`* argument to `mysqld`. For example, if you specified `/backups/binarylog` as your base name, you will find a series of logs in */backups* named *binarylog.001*, *binarylog.002*, and so on.

Tip ✎

The typical practice for the binary log is to specify as the base name the same directory as `datadir`, where database files are kept. I prefer to store this important log in another directory, perhaps on another filesystem, along with the output from `mysqldump` or any other backup utilities.

There are two ways to apply the binary logfiles to a restored database. The first is to simply apply them directly without converting them to text. The second option is to convert them to text, possibly edit the text file, and apply them as SQL commands.

The first thing that you do, of course, is restore the database using one of the methods covered earlier in the chapter. Use `mysqldump`, `mysqlhotcopy`, or your own custom method to back up and recover the database in question. Once that is done, the database has been restored to the point in time the backup was taken. What you need then are all binary logs that were modified since that point.

Directly applying binary logs

The simplest method of applying binary logs to an already restored database is to pipe them into `mysql`. Suppose, for example, you had two binary logs that have been modified since your last backup, and they are called */backups/binarylog.093* and */backups/binarylog.094*. All you have to do is tell the `mysqlbinlog` command to process the logs, and to pass the results of that process directly to `mysql`:

```
$ mysqlbinlog --database=database/backups/binarylog.093
/backups/binarylog.094|mysql
```

This command passes to `mysqlbinlog`'s output (and `mysql`'s input) a series of SQL statements that reflect changes that were made to the specified database since the point in time that the database was restored.

Applying binary logs via temporary SQL files

Since `mysqlbinlog` sends to standard output the SQL commands it finds in each log, you can also create a

temporary file that contains those SQL commands using this command:

```
$ mysqlbinlog --database=database /backups/binarylog.093 \/backups/binarylog.094
>/backups/myredo.sql
```

That temporary file can then be passed to `mysql`, resulting in the SQL statements being reexecuted:

```
$ mysql < /backups/myredo.sql
```

These two commands accomplish the same thing as the single command in the previous section, but they also allow you to edit the SQL file if you so desire.

Applying binary logs using date ranges

Sometimes you don't want the database restored all the way to the point of failure, especially when the "failure" was a human one. For example, suppose a DBA accidentally dropped a very important table, and you were not able to roll that transaction back. (If you use MyISAM tables, for example, you *can't* roll transactions back.)

What you can do is restore the database to the last full backup, then apply the binary logs until right before you dropped the table. There are two ways to accomplish this. If you know exactly when the `drop table` command was issued, you can pass a date range to `mysqlbinlog`:

```
$ mysqlbinlog --start-date="2006-07-09 15:00:00" --stop-date="2006-07-10
15:00:00" \/backups/binarylog.093 /backups/binarylog.094 |mysql
```

Another way would be to use the command from the previous section that creates a text file containing all the SQL statements from the bin log:

```
$ mysqlbinlog /backups/binarylog.093 /backups/binarylog.094 >/backups/myredo.sql
```

Look at the text file */backups/myredo.sql* to determine the position number of the `drop table` command. (Position

numbers are logged in the binary log starting with the string `log_pos`.) Identify the position number of the bad command, and tell `mysqlbinlog` to stop before that position:

```
$ mysqlbinlog --start-position="337280" --stop-position="337302"
\/backups/binarylog.093 /backups/binarylog.094 |mysql
```

Finally, you can also create the text file with all the SQL commands and simply edit out the one command you don't like. You can then pass the remaining SQL statements to `mysql`. First create the file:

```
$ mysqlbinlog /backups/binarylog.093 /backups/binarylog.094 >/backups/myredo.sql
```

Edit it with the text editor of your choice, and pass the remaining commands to `mysql`:

```
$ cat /backups/myredo.sql|mysql
```

MySQL Cluster Hot Backup and Recovery

Since MySQL cluster databases actually reside on several nodes, backups work a bit differently. A single command causes each cluster node to write three files to a specified directory. Once you've received notification that the backup is complete, you can back up or copy the three files using any method you wish.

As mentioned previously, the backup consists of three files that contain different types of data. The names of these backup files start with *BACKUP* and contain the *<backup_id>* and *<node_id>* to which that file belongs. The *<backup_id>*is a unique identifier for each backup that is assigned by the backup process, and *<node_id>* is the unique identifier for each node that creates a backup file.

Metadata is stored in the *BACKUP-<backup_id>.<node_id>.ctl* file.

This backup file contains the names and definitions of all database tables. Each node backs up the definitions for all tables in the cluster to its metadata backup file.

Table records are stored in the *BACKUP-<backup_id>-0.<node_id>.datafile*.

This backup file contains the actual data stored in the database (at the time the backup was made). Each node stores a fragment of the entire database, and the backup file starts with a header specifying the tables to which the records in a given backup file belong.

The transaction log is stored in the *BACKUP-<backup_id>.<node_id>.logfile*.

This backup file contains a transactional record specifying how data was stored in the database. Each node backs up only the transaction log for the tables that it backed up in the table records backup file.

Initial configuration

If you want to perform a MySQL cluster backup, there are four configuration parameters in MySQL. Their default values typically work for most environments.

`BackupDataBufferSize`

The amount of memory used when writing data to disk.

`BackupLogBufferSize`

The amount of memory that should be used for log records before they’re written to disk.

`BackupMemory`

The total amount of memory assigned to backup; it should be the sum of `BackupDataBufferSize` and `BackupLogBufferSize`.

`BackupWriteSize`

The block size of data written to disk, which applies to all parts of the backup.

Optionally, you can specify a value for `BackupDataDir` to specify that all cluster backup files be sent to this directory. One idea is to use an NFS directory and send there all backup datafiles using this parameter. If you do not specify a value for `BackupDataDir`, backups are placed in a subdirectory called *BACKUP* in the location specified by the `FileSystemPath` parameter.

Performing a backup of MySQL cluster

Performing a backup is relatively easy: you essentially have to run only one command and watch for its output:

1. Start the backup by issuing this command at the shell prompt:
2.
```
ndb_mgm –e "start backup"
```
3. Once the management client has submitted a request to the cluster to start the backup, it responds with the phrase `Start of backup ordered`. At this point it has submitted only a request, and it has not received a response from the cluster.
4. Once the backup actually starts, the management client responds that `Backup` *`backup_id`* `started`, where *`backup_id`* is the unique identifier assigned to this backup. As mentioned previously, this unique identifier is used as part of the name of the backup files. At this point, the backup has started, but it has not finished.

Tip ✎

This backup may take a while to complete. If you want to cancel a backup you have started, enter the command:

```
ndb_mgm –e "abort backup backup_id"
```

where `backup_id` is the unique identifier you were previously given.

1. Once the backup is completed, the management client responds with the message `Backup` *`backup_id`* `completed`.
2. At this point, each node has created the three files mentioned earlier in the directory you previously specified.

Restoring a MySQL cluster

MySQL cluster backups are restored with the backups mentioned previously using the `ndb_restore` command. The command must be executed once for each set of three backup files. In other words, it must be run as many times as there are nodes in the cluster.

To perform a MySQL cluster restore, the cluster must be operational, and you should have an empty database to restore to. There must also be a free connection to the cluster to perform the restore. This can be verified by issuing the `ndb_mgm –e show` command.

The following steps can be used to restore a MySQL cluster database:

1. Run the command `ndbd --initial` on all storage nodes.
2. Place the cluster in single user mode.
3. Once you have an empty database to restore to and have verified that you can connect to the database, it's time to restore it.
4. Run `ndb_restore` and tell it to restore the first node. Specify the host and port number of the MySQL Cluster Management Server with the `–c` *`mgm_host:port`* option, followed by the first node in the cluster (for example, `–n 1`).

> **Tip ✎**
>
> You can actually restore your nodes in any order. Starting with the first node just keeps it orderly. Just remember to specify the `–m` option for the first node.

1. You also need to specify a `backup_id` number `–b` *`backup_id`* (for example, `–b 7`.) and the name of the directory where you created the backup files (for example, */ backups/mysqlcluster*). The first time you run `ndb_restore`, you have to specify the `–m` option to rebuild the tables. Here is an example `ndb_restore` command to restore the first node in the cluster from `backup_id 7`, with the backups located in */backups/mysqlcluster*:
2.
```
$ ndb_restore -c mgm_host:port -n 1 -m –b 7 -r /backups/mysqlcluster
```

3. Once that's done, repeat the command for each node in the cluster. The following code is for a four-node cluster. Since you already restored the first node in the previous step, there are three more nodes to restore.

4.
```
$ ndb_restore -c mgm_host:port -n 2 -b 7 -r /backups/mysqlcluster
$ ndb_restore -c mgm_host:port -n 3 -b 7 -r /backups/mysqlcluster
$ ndb_restore -c mgm_host:port -n 4 -b 7 -r /backups/mysqlcluster
```

Once you've successfully executed the first `ndb_restore`, you can run the other restores in parallel if you wish.

> **Tip ✎**
>
> BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Part 6. Potpourri

Part VI consists of two chapters:

Chapter 23, VMware and Miscellanea

Covers a variety of topics ranging from how to back up VMware servers to dealing with volatile filesystems. This chapter also offers some backup poetry to inspire you.

Chapter 24, It’s All About Data Protection

Explains that backup and recovery is just one piece of data protection and describes other important considerations including disaster recovery and how to assess your data protection needs.

Chapter 23. VMware and Miscellanea

No matter how we organized this book, there would be subjects that wouldn't fit anywhere else. This chapter covers these subjects, including important information such as backing up volatile filesystems and handling the difficulties inherent in gigabit Ethernet.

Backing Up VMware Servers

The popularity of VMware virtual servers has grown significantly in the last few years, prompting questions on how to back them up. First, we'll describe the architecture of VMware and follow that with a discussion of how to back it up.

VMware Architecture

VMware currently comes in two basic flavors, VMware Server and VMware ESX Server. VMware Server is a free version of VMware that offers basic virtual server capabilities and runs inside Linux or Windows. Each virtual machine is represented as a series of files in a subdirectory of a standard filesystem that you specify; the subdirectory carries the name of the virtual machine. For example, if you've chosen to store your virtual machines in */vmachines*, and you have a virtual host called Windows 2000, its files will be located in */vmachines/Windows 2000*.

While VMware Server runs inside standard Linux or Windows, VMware ESX Server uses a custom Linux kernel and a custom filesystem, VMFS, to store virtual machine files. You can also store virtual machine files on raw disk partitions. Neither the raw disk partitions or files in a

VMFS filesystem can be accessed by all backup commands, so you probably need to back them up in a special way.

VMware Backups

When backing up a VMware machine, you have to back up the operating system of the VMware server (known as the *service console* on ESX systems) and the VMware application itself. You also need to back up each virtual machine's files. However, you cannot simply back up the virtual machine files or raw disks with a standard backup program. The virtual disks are constantly open and changing while the virtual machines are running, and you will not receive a consistent backup of them. Even open file agents won't necessarily work properly if the virtual machines are gigabytes in size. You therefore have three options for backing up virtual machines running inside VMware:

- Back up virtual machines as physical machines.
- Back up virtual files while virtual machines are suspended.
- Use VMware's built-in tools to copy a running virtual machine's files.

Back up virtual machines as physical machines

This is, of course, the easy method. Simply pretend that each virtual machine is a physical machine, and back it up as such. This method has both advantages and disadvantages.

> **Tip ✎**
>
> If you use this method, don’t forget to exclude the virtual machine files when you’re backing up the VMware Server or ESX service console.

The first advantage of this method is that you can use the same backup system as the rest of your data center. Just because the machines are virtual doesn’t mean you have to treat them as such. This simplifies your backup system. It also allows you to take advantage of full and incremental backups. Unless your backup software is able to perform subfile incremental backups, the other two methods perform full backups every day because the entire virtual machine is represented by a single file that will certainly change every day. Finally, it allows you to back up the systems live.

The disadvantage is that you have to configure backups for each virtual machine. Some may prefer to configure one backup for the entire VMware server. If you’re using a commercial backup software package, this also increases your cost because you must buy a license for each virtual machine. A final disadvantage is that you also need to configure a bare-metal recovery backup for each virtual machine. Each of the other two methods won’t need such a backup because restoring the virtual machine is all that’s needed to perform its bare-metal recovery.

Back up suspended virtual machine files

If you can afford the downtime for each virtual machine, all you have to do is suspend a virtual machine prior to backing up its files. You can then back up the virtual machine files using your favorite backup program because

the files won't be open or changing during your backup. The suspend function in VMware works just the same as on your laptop (and some servers). The current memory image and running processes are saved to a file that is then accessed when you power it on, causing all running processes to resume where they left off just before you suspended the machine.

The advantages to this method are that you don't need to configure backups for every virtual machine, and you don't need to worry about bare-metal recovery of these machines. The first disadvantage is that it performs a full backup of each virtual server every night, unless you have a backup that can perform subfile incrementals. The only open-source product that may be able to do that is *rdiff-backup*; see Chapter 7 for more information. The second disadvantage is that it requires suspending virtual machines, which renders them unusable during the backup. This downtime may be undesirable in many environments.

> **Tip ✎**
>
> One way to have your cake and eat it too is if your virtual machines are residing on an LVM volume that supports snapshots. You could suspend all the virtual machines, take an LVM snapshot, then unsuspend the virtual machines. This minimizes the amount of time the virtual machine is suspended but allows you to take as long as you need to back up the LVM snapshot.

To suspend a running machine from the command line, run the following command, where the *.vmx* file is the configuration file stored in the virtual machine directory:

```
C:> vmware-cmd path-to-config/config.vmx suspend
```

You can now back up the virtual files using any method you choose. After backups have completed, you can restart the machine with the following command:

```
C:> vmware-cmd path-to-config/config.vmx start
```

> **Tip ✎**
>
> If you use ESX Server, your backup tools may have trouble accessing the VMFS files. Make sure you test any new backup method.

Copy/export a running virtual machine using VMware's tools (ESX only)

Finally, if you're running VMware ESX Server, you can use the Perl APIs to copy the virtual machine files while the machine is running. They do this by creating a snapshot of the changes that happen to the virtual machine during the backup, storing them in a redo log, then copying or exporting the virtual files to another location. This method has the same advantages mentioned for the previous method, and it comes with the additional advantage of being able to back up systems while they're running.

This process requires using ever-changing Perl scripts, so we won't cover implementation details here. The VMware web site (http://www.vmware.com) includes some example scripts. There's also an open-source tool called `vmbk`, written by Massimiliano Daneri, that can automate this process. Learn more about it at http://www.vmts.net/vmbk.htm.

Using Bare-Metal Recovery to Migrate to VMware

One of the really nice things about using VMware (or other virtual server solutions) is that you don't have worry about bare-metal recovery of the virtual servers. As long as you can get them to not change during a backup, all you have to do is back up their files.

However, you can use the bare-metal recovery procedure documented in Chapter 11 to migrate physical machines into virtual machines. We just did this and turned 25 very old physical servers into one very nice VMware server. The following is the story of that migration.

I get asked all kinds of questions about backup products and how they behave on different operating systems and applications, and I use a lab to answer these questions. In addition to the usual backup hardware (SAN, tape libraries, VTLs), it consists of some Sun, IBM, and HP hardware running Solaris, AIX, and HP-UX. Up until just recently, we also had about 25 Intel machines running various versions of Linux and Windows and their associated applications (Exchange, SQL Server, Oracle, etc.). I never had enough machines, and I never had the right machines connected to the right hardware. We were constantly swapping SCSI and Fibre Channel cards, as well as installing and uninstalling applications. I could have used 100 machines, but that would obviously be prohibitive in many ways. (The cooling alone would be crazy.)

So we recently decided to see if we could get rid of all these servers with VMware. We bought a white box with a 3.5 GHz Dual Core AMD processor, 4 GB DDR2 RAM and 1.75 TB of internal SATA disks. I installed into that server two existing Fibre Channel cards and two SCSI cards. I then followed the alt-boot recovery method to move all of those physical servers into virtual servers, virtually upgrading each of their CPUs, storage, and memory in the process. Here are the steps I followed for each server:

1. I used the alt-boot full image method to create an image of the entire */dev/hda* hard drive to an NFS mount on the new VMware server. (These images were typically 4–10 GB. They were old servers!)
2. I used VMware to create a virtual machine specifying a virtual IDE hard drive that was much bigger than the original, usually about 20 or 40 GB.
3. I used VMware to create a virtual CD drive that pointed to an ISO file that was actually a symbolic link to an ISO image of a Knoppix CD on the hard drive.
4. I booted the virtual machine into Knoppix using the virtual Knoppix CD.
5. I used `dd` to copy the image of the real hard drive to the virtual hard drive in the virtual machine booted from the virtual CD. (We did this by mounting the NFS drive where we stored the image.)
6. I “removed” the Knoppix CD by changing the symbolic link to point to an ISO image of a nonbootable CD and rebooted the virtual server.
7. In almost every case, the virtual server came up without incident, and voila! I had moved a physical server into a virtual server without a hitch! One Windows server was blue-screening during the boot, but I pressed F8 and selected Last Known Good Configuration, and it booted just fine.
8. I installed VMware tools into each virtual machine, which made their video and other drivers much happier.
9. Once I verified the health of each machine, I changed the CD symbolic link to point to Knoppix again and booted into Knoppix. I then used either `qtparted` (for Linux systems) or `fdisk` and `ntfsresize` (for Windows

systems) to grow the original hard drive to the new size, as discussed in the sidebar "Restoring to Larger Hard Drives" in Chapter 11.

With 4 GB of RAM and a 3.5 GHz dual-core processor, I can run about eight virtual servers at a time without swapping. I typically only need a few at a time, and what's important is that I have Exchange 2000, SQL Server X, or XYZ x.x running; they don't need to run that fast. (That's how I was able to get by with those old servers for so long.) Each virtual server can have access to either one of the Fibre Channel cards or SCSI cards, which gives them access to every physical and virtual tape drive in the lab. They will also have more CPU, disk, and RAM than they ever had in their old machine. (I can even temporarily give any of them the entire 3.5 GHz processor and almost all of the 4 GB of RAM if I need to, and I don't have to swap chips or open up any CPU thermal compound to do it!)

I also get to have hundreds of virtual servers and not have any logistical or cooling issues, since each server only represents 20–50 GB of space on the hard drive. I can have a Windows 2000 server running no special apps, one running Exchange 5, one running SQL Server 7, a server running Windows 2003 with no special apps, one with Exchange 2000, and one running Windows Vista. I could have servers running every distribution of Linux, FreeBSD, and Solaris x86—and all of the applications those servers support. I think you get the point. I've got enough space for about 300 virtual server combinations like that. It boggles the mind.

Volatile Filesystems

A volatile filesystem is one that changes almost constantly while it is being backed up. Backing up a volatile filesystem could result in a number of negative side effects. The degree to which a backup is affected is directly proportional to the volatility of the filesystem and highly dependent on the backup utility that you are using. Some files could be missing or corrupted within the backup, or the wrong versions of files may be found within the backup. The worst possible problem, of course, is that the backup itself could become corrupted, although this should happen only under the most extreme circumstances. (See the section “Demystifying dump” for details on what can happen when performing a `dump` backup of a volatile filesystem.)

Missing or Corrupted Files

Files that are changing during the backup do not always make it to the backup correctly. This is especially true if the filename or inode changes during the backup. The extent to which your backup is affected by this problem depends on what type of utility you’re using and how volatile the filesystem is.

For example, suppose that the utility performs the equivalent of a `find` command at the beginning of the backup. This utility then begins backing up those files based on the list that it created at the beginning of the backup. If a filename changes during a backup, the backup utility receives an error when it attempts to back up the old filename. The file, with its new name, is simply overlooked.

Another scenario occurs when the filename does not change, but its contents do. The backup utility begins backing up the file, and the file changes while being backed up. This is probably most common with a large database file. The backup of this file would be essentially worthless because different parts of it were created at different times. (This is actually what happens when backing up Oracle database files in hot-backup mode. Without Oracle's ability to rebuild the file, the backup of these files would be worthless.)

Referential Integrity Problems

This is similar to the corrupted files problem but on a filesystem level. Backing up a particular filesystem may take several hours. This means that different files within the backup are backed up at different times. If these files are unrelated, this creates no problem. However, suppose that two different files are related in such a way that if one is changed, the other is changed. An application needs these two files to be related to each other. This means that if you restore one, you must restore the other. It also means that if you restore one file to 11:00 p.m. yesterday, you should restore the other file to 11:00 p.m. yesterday. (This scenario is most commonly found in databases but can be found in other applications that use multiple, interrelated files.)

Suppose that last night's backup began at 10:00 p.m. Because of the name or inode order of the files, one is backed up at 10:15 p.m. and the other at 11:05 p.m. However, the two files were changed together at 11:00 p.m., between their separate backup times. Under this scenario, you would be unable to restore the two files to the way they looked at any single point in time. You could restore the first file to how it looked at 10:15, and the

second file to how it looked at 11:05. However, they need to be restored together. If you think of files within a filesystem as records within a database, this would be referred to as a referential integrity problem.

Corrupted or Unreadable Backup

If the filesystem changes significantly while it is being backed up, some utilities may actually create a backup that they cannot read. This is obviously one of the most dangerous things that can happen to a backup, and it would happen only under the most extreme circumstances.

Torture-Testing Backup Programs

In 1991, Elizabeth Zwicky did a paper for the LISA[1] conference called “Torture-testing Backup and Archive Programs: Things You Ought to Know But Probably Would Rather Not.” Although this paper and its information are somewhat dated now, people still refer to it when talking about this subject. Elizabeth graciously consented to allow us to include some excerpts in this book:

Many people use `tar`, `cpio`, or some variant to back up their filesystems. There are a certain number of problems with these programs documented in the manual pages, and there are others that people hear of on the street, or find out the hard way. Rumors abound as to what does and does not work, and what programs are best. I have gotten fed up, and set out to find Truth with only `Perl` (and a number of helpers with different machines) to help me.

As everyone expects, there are many more problems than are discussed in the manual pages. The rest of the results are startling. For instance, on Suns running SunOS 4.1, the manual pages for both `tar` and `cpio` claim bugs that the

programs don’t actually have any more. Other “known’’ bugs in these programs are also mysteriously missing. On the other hand, new and exciting bugs—bugs with symptoms like confusions between file contents and their names—appear in interesting places.

Elizabeth performed two different types of tests. The first type were static tests that tried to see which types of programs could handle strangely named files, files with extra-long names, named pipes, and so on. Since at this point we are talking only about volatile filesystems, I will not include her static tests here. Her active tests included:

- A file that becomes a directory
- A directory that becomes a file
- A file that is deleted
- A file that is created
- A file that shrinks
- Two files that grow at different rates

Elizabeth explains how the degree to which a utility would be affected by these problems depends on how that utility works:

Programs that do not go through the filesystem, like `dump` , write out the directory structure of a filesystem and the contents of files separately. A file that becomes a directory or a directory that becomes a file will create nasty problems, since the content of the inode is not what it is supposed to be. Restoring the backup will create a file with the original type and the new contents.

Similarly, if the directory information is written out and then the contents of the files, a file that is deleted during the run will still appear on the volume, with indeterminate

contents, depending on whether or not the blocks were also reused during the run.

All of the above cases are particular problems for `dump` and its relatives; programs that go through the filesystem are less sensitive to them. On the other hand, files that shrink or grow while a backup is running are more severe problems for `tar` , and other filesystem based programs. `dump` will write the blocks it intends to, regardless of what happens to the file. If the block has been shortened by a block or more, this will add garbage to the end of it. If it has lengthened, it will truncate it. These are annoying but nonfatal occurrences. Programs that go through the filesystem write a file header, which includes the length, and then the data. Unless the programmer has thought to compare the original length with the amount of data written, these may disagree. Reading the resulting archive, particularly attempting to read individual files, may have unfortunate results.

Theoretically, programs in this situation will either truncate or pad the data to the correct length. Many of them will notify you that the length has changed, as well. Unfortunately, many programs do not actually do truncation or padding; some programs even provide the notification anyway. (The “`cpio` out of phase: get help!” message springs to mind.) In many cases, the side reading the archive will compensate, making this hard to catch. SunOS 4.1 `tar`, for instance, will warn you that a file has changed size, and will read an archive with a changed size in it without complaints. Only the fact that the test program, which runs until the archiver exits, got ahead of `tar`, which was reading until the file ended, demonstrated the problem. (Eventually the disk filled up, breaking the deadlock.)

Other warnings

Most of the things that people told me were problems with specific programs weren't; on the other hand, several people (including me) confidently predicted correct behavior in cases where it didn't happen. Most of this was due to people assuming that all versions of a program were identical, but the name of a program isn't a very good predictor of its behavior. Beware of statements about what `tar` does, since most of them are either statements about what it ought to do, or what some particular version of it once did....Don't trust programs to tell you when they get things wrong either. Many of the cases in which things disappeared, got renamed, or ended up linked to fascinating places involved no error messages at all.

Conclusions

These results are in most cases stunningly appalling. `dump` comes out ahead, which is no great surprise. The fact that it fails the name length tests is a nasty surprise, since theoretically it doesn't care what the full name of a file is; on the other hand, it fails late enough that it does not seem to be an immediate problem. Everything else fails in some crucial area. For copying portions of filesystems, `afio` appears to be about as good as it gets, if you have long filenames. If you know that all of the files will fit within the path limitations, GNU `tar` is probably better, since it handles large numbers of links and permission problems better.

There is one comforting statement in Elizabeth's paper: "It's worth remembering that most people who use these programs don't encounter these problems." Thank goodness!

Using Snapshots to Back Up a Volatile Filesystem

What if you could back up a very large filesystem in such a way that its volatility was irrelevant? A recovery of that filesystem would restore all files to the way they looked when the entire backup began, right? A technology called the *snapshot* allows you to do just that. A snapshot provides a *static view* of an active filesystem. If your backup utility is viewing a filesystem through its snapshot, it could take all night long to back up that filesystem; yet it would be able to restore that filesystem to exactly the way it looked when the entire backup began.

How do snapshots work?

When you create a snapshot, the software records the time at which the snapshot was taken. Once the snapshot is taken, it gives you and your backup utility another name through which you may view the filesystem. For example, when a Network Appliance creates a snapshot of */home*, the snapshot may be viewed at */home/.snapshot*. Creating the snapshot doesn't actually copy data from */home* to */home/.snapshot*, but it appears as if that's exactly what happened. If you look inside */home/.snapshot*, you'll see the entire filesystem as it looked at the moment when */home/.snapshot* was created.

Actually creating the snapshot takes only a few seconds. Sometimes people have a hard time grasping how the software could create a separate view of the filesystem without copying it. This is why it is called a snapshot: it didn't actually copy the data, it merely took a "picture" of it.

Once the snapshot has been created, the software monitors the filesystem for activity. When it is going to change a block of data, it keeps a record of the way the block used to look like when you took the snapshot. How a given product holds on to the previous image varies from product to product.

When you view the filesystem using the snapshot directory, it watches what you're looking for. If you request a block of data that has not changed since the snapshot was taken, it retrieves that block from the actual filesystem. However, if you request a block of data that has changed since the snapshot was taken, it retrieves that block from another location. This, of course, is completely invisible to the user or application accessing the data. The user or application simply views the filesystem using the snapshot, and where the blocks come from is managed by the snapshot software.

Available snapshot software

Most NAS filers now have built-in snapshot capabilities. In addition, many filesystem drivers now support the ability to create snapshots as well.

---

[1] Large Installation System Administration Conference, sponsored by Usenix and Sage (http://www.usenix.org).

Demystifying dump

> **Tip ✎**
>
> This section was written by David Young.

`cpio` , `ntbackup`, and `tar` are filesystem-based utilities, meaning that they access files through the filesystem. If a backup file is changed, deleted, or added during a backup, usually the worst thing that can happen is that the contents of the individual file that changed will be corrupt. Unfortunately, there is one huge disadvantage to backing up files through the filesystem: the backup affects inode times (`atime` or `ctime`).

`dump`, on the other hand, does *not* access files though the Unix filesystem, so it doesn't have this limitation. It backs up files by accessing the data through the raw device driver. Exactly how `dump` does this is generally a mystery to most system administrators. The `dump` manpage doesn't help matters either, since it creates FUD (fear, uncertainty, and doubt). For example, Sun's `ufsdump` manpage says:

When running `ufsdump`, the filesystem must be inactive; otherwise, the output of `ufsdump` may be inconsistent and restoring files correctly may be impossible. A filesystem is inactive when it is unmounted [sic] or the system is in single user mode.

From this warning, it is not very clear the extent of the problem if the advice is not heeded. Is it individual files in the dump that may be corrupted? Is it entire directories? Is it everything beyond a certain point in the dump? Is it the entire dump? Do we *really* have to dismount the filesystem to get a consistent dump?

Questions like these raise a common concern when performing backups with dump. Will we learn (after it's too late) that a backup is corrupt just because we dumped a mounted filesystem, even though it was essentially idle at the time? If we are going to answer these questions, we need to understand exactly how dump works.

Dumpster Diving

The dump utility is very filesystem-specific, so there may be slight variations in how it works on various Unix platforms. For the most part, however, the following description should cover how it works because most versions of dump are generally derived from the same code base. Let's first look at the output from a real dump. We're going to look at an incremental backup because it has more interesting messages than a level-0 backup:

```
# /usr/sbin/ufsdump 9bdsfnu 64 80000 150000 /dev/null /
  DUMP: Writing 32 Kilobyte records
  DUMP: Date of this level 9 dump: Mon Feb 15 22:41:57 2006
  DUMP: Date of last level 0 dump: Sat Aug 15 23:18:45 2005
  DUMP: Dumping /dev/rdsk/c0t3d0s0 (sun:/) to /dev/null.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 56728 blocks (27.70MB) on 0.00 tapes.
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 56638 blocks (27.66MB) on 1 volume at 719 KB/sec
  DUMP: DUMP IS DONE
  DUMP: Level 9 dump on Mon Feb 15 22:41:57 2006
```

In this example, ufsdump makes four main passes to back up a filesystem. We also see that Pass II was performed three times. What is dump doing during each of these passes?

Pass I

Based on the entries in the *dumpdates* file (usually */etc/dumpdates*) and the dump level specified on the command line, an internal variable named `DUMP_SINCE` is calculated. Any file modified after the `DUMP_SINCE` time is a candidate for the current dump. `dump` then scans the disk and looks at all inodes in the filesystem. Note that `dump` "understands" the layout of the Unix filesystem and reads all of its data through the raw disk device driver.

Unallocated inodes are skipped. The modification times of allocated inodes are compared to `DUMP_SINCE`. Modification times of files greater than or equal to `DUMP_SINCE` are candidates for backup; the rest are skipped. While looking at the inodes, `dump` builds:

- A list of *file* inodes to back up
- A list of *directory* inodes seen
- A list of *used* (allocated) inodes

Pass IIa

`dump` rescans all the inodes and specifically looks at directory inodes that were found in Pass I to determine whether they contain any of the files targeted for backup. If not, the directory's inode is dropped from the list of directories that need to be backed up.

Pass IIb

By deleting in Pass IIa directories that do not need to be backed up, the parent directory may now qualify for the same treatment on this or a later pass, using this algorithm. This pass is a rescan of all directories to see if the remaining directories in the directory inode list now qualify for removal.

Pass IIc

Directories were dropped in Pass IIb. Perform another scan to check for additional directory removals. This ends up being the final Pass II scan because no more directories can be dropped from the directory inode list. (If additional directories had been found that could be dropped, another Pass II scan would have occurred.)

Pre-Pass III

This is when `dump` actually starts to write data. Just before Pass III officially starts, `dump` writes information about the backup. `dump` writes all data in a very structured manner. Typically, `dump` writes a header to describe the data that is about to follow, and then the data is written. Another header is written and then more data. During the Pre-Pass III phase, `dump` writes a dump header and two inode maps. Logically, the information would be written sequentially, like this:

header

    `TS_TAPE` dump header

header

    `TS_CLRI`

usedinomap

    A map of inodes deleted since the last dump

header

    `TS_BITS`

dumpinomap

    A map of inodes in the dump

The map *usedinomap* is a list of inodes that have been deleted since the last dump. `restore` uses this map to delete files before doing a restore of files in this dump. The map

dumpinomap is a list of all inodes contained in this dump. Each header contains quite a bit of information:

Record type
Dump date
Volume number
Logical block of record
Inode number
Magic number
Record checksum
Inode
Number of records to follow
Dump label
Dump level
Name of dumped filesystem
Name of dumped device
Name of dumped host
First record on volume

The record type field describes the type of information that follows the header. There are six basic record types:

`TS_TAPE`

dump header

`TS_CLRI`

Map of inodes deleted since last dump

`TS_BITS`

Map of inodes in dump

`TS_INODE`

Beginning of file record

`TS_ADDR`

Continuation of file record

`TS_END`

End of volume marker

It should be noted that when dump writes the header, it includes a copy of the inode for the file or directory that immediately follows the header. Since inode data structures have changed over the years, and different filesystems use slightly different inode data structures for their respective filesystems, this would create a portability problem. So dump normalizes its output by converting the current filesystem's inode data structure into the old BSD inode data structure. It is this BSD data structure that is written to the backup volume.

As long as all dump programs do this, then you should be able to restore the data on any Unix system that expects the inode data structure to be in the old BSD format. It is for this reason you can interchange a dump volume written on Solaris, HP-UX, and AIX systems.

Pass III

This is when real disk data starts to get dumped. During Pass III, dump writes only those directories that contain files that have been marked for backup. As in the Pre-Pass III phase, during Pass III, dump logically writes data something like this:

Header (TS_INODE)
Disk blocks (directory block[s])
Header (TS_ADDR)
Disk blocks (more directory block[s])
.
.
.
Header (TS_ADDR)
Disk blocks (more directory block[s])
Repeat the previous four steps for each directory in the list of directory inodes to back up

Pass IV

Finally, file data is dumped. During Pass IV, dump writes only those files that were marked for backup. dump logically writes data during this pass as it did in Pass III for directory data:

Header (TS_INODE)
Disk blocks (file block[s])
Header (TS_ADDR)
Disk blocks (more file block[s])
.
.
.
Header (TS_ADDR)
Disk blocks (more file block[s])
Repeat the previous four steps for each file in the list of file inodes to back up.

Post-Pass IV

To mark the end of the backup, dump writes a final header using the TS_END record type. This header officially marks the end of the dump.

Summary of dump steps

The following is a summary of each of dump's steps:

Pass I

dump builds a list of the files it is going to back up.

Pass II

dump scans the disk multiple times to determine a list of the directories it needs to back up.

Pre-Pass III

dump writes a dump header and two inode maps.

Pass III

`dump` writes a header (which includes the directory inode) and the directory data blocks for each directory in the directory backup list.

Pass IV

`dump` writes a header (which includes the file inode), and the file data blocks for each file in the file backup list.

Post-Pass IV

`dump` writes a final header to mark the end of the dump.

Answers to Our Questions

Let's review the issues raised earlier in this section.

Question 1

Q: If we dump an active filesystem, will data corruption affect individual directories/files in the dump?

A: Yes.

The following is a list of scenarios that can occur if your filesystem is changing during a dump:

A file is deleted before Pass I

The file is not included in the backup list because it doesn't exist when Pass I occurs.

A file is deleted after Pass I but before Pass IV

The file may be included in the backup list, but during Pass IV, `dump` checks to make sure the file still exists and is a file. If either condition is `false`, `dump` skips backing it up. However, the inode map written in Pre-Pass III will be incorrect. This inconsistency does not affect the dump, but `restore` will be unable to recover the file even though it is in the restore list.

The contents of a file marked for backup change (inode number stays the same); there are really two scenarios here

Changing the file at a time when dump is *not* backing it up does not affect the backup of the file. dump keeps a list of the inode numbers, so changing the file may affect the contents of the inode but not the inode number itself.

Changing the file when dump is backing up the file probably will corrupt the data dumped for the current file. dump reads the inode and follows the disk block pointers to read and then write the file blocks. If the address or contents of just one block changes, the file dumped will be corrupt.

The inode number of a file changes

If the inode number of a file changes after it was put on the backup list (inode changes after Pass I, but before Pass IV), then when the time comes to back up the file, one of three scenarios occurs:

The inode is not being used by the filesystem, so dump will skip backing up this file. The inode map written in Pre-Pass III will be incorrect. This inconsistency will not affect the dump but will confuse you during a restore (a file is listed but can’t be restored).

The inode is reallocated by the filesystem and is now a directory, pipe, or socket. dump will see that the inode is not a regular file and ignore the backing up of the inode. Again, the inode map written in Pre-Pass III will be inconsistent.

The inode is reallocated by the filesystem and now is used by another file; dump will back up the new file. Even worse, the name of the file dumped in Pass III for that inode number is incorrect. The file actually may be of a file somewhere else in the filesystem. It’s like dump trying to back up */etc/hosts* but really getting */bin/ls*. Although the file is not corrupt in the true sense of the word, if this file were restored, it would not be the correct file.

A file is moved in the filesystem.

Again, there are a few scenarios:

The file is renamed before the directory is dumped in Pass III. When the directory is dumped in Pass III, the new name of the file will be dumped. The backup then proceeds as if the file was never renamed.

The file is renamed after the directory is dumped in Pass III. The inode doesn't change, so `dump` will back up the file. However, the name of the file dumped in Pass III will not be the current filename in the filesystem. This scenario should be harmless.

The file is moved to another directory in the same filesystem before the directory was dumped in Pass III. If the inode didn't change, then this is the same as the first scenario.

The file is moved to another directory in the same filesystem after the directory was dumped in Pass III. If the inode didn't change, then the file will be backed up, but during a restore it would be seen in the old directory with the old name.

The file's inode changes. The file would not be backed up, or another file may be backed up in its place (if another file has assumed this file's old inode).

Question 2

Q: If we dump an active filesystem, will data corruption affect directories?

A: Possibly.

Most of the details outlined for files also apply to directories. The one exception is that directories are dumped in Pass III instead of Pass IV, so the time frames for changes to directories will change.

This also implies that changes to directories are less susceptible to corruption because the time that elapses between the generation of the directory list and the dump of that list is less. However, changes to files that normally would cause corresponding changes to the directory information still will create inconsistencies in the dump.

Question 3

Q: If we dump an active filesystem, will data corruption affect the entire dump or everything beyond a certain point in the dump?

A: No.

Even though `dump` backs up files through the raw device driver, it is in effect backing up data inode by inode. This is still going through the filesystem and doing it file by file. Corrupting one file will not affect other files in the dump.

Question 4

Q: Do we really *have to dismount the filesystem to get a consistent dump?*

A: No.

There is a high likelihood that dumps of an idle, mounted filesystem will be fine. The more active the filesystem, the higher the risk that corrupt files will be dumped. The risk that files are corrupt is about the same for a utility that accesses files using the filesystem.

Question 5

Q: Will we learn (after it's too late) that our dump of an essentially idle mounted filesystem is corrupt?

A: No.

It's possible that individual files in that dump are corrupt, but highly unlikely that the entire dump is corrupt. Since `dump` backs up data inode by inode, this is similar to backing up through the filesystem file by file.

A Final Analysis of dump

As described earlier, using `dump` to back up a mounted filesystem can dump files that are found to be corrupt when

restored. The likelihood of that occurring rises as the activity of the filesystem increases. There are also situations that can occur where data is backed up safely, but the information in the dump is inconsistent. For these inconsistencies to occur, certain events have to occur at the right time during the dump. And it is possible that the wrong file is dumped during the backup; if that file is restored, the administrator will wonder how that happened!

> **Tip ✎**
>
> Be sure to read the sidebar “dump on Mac OS and Linux” in Chapter 3.

The potential for data corruption to occur is pretty low but still a possibility. For most people, dumping live filesystems that are fairly idle produces a good backup. Generally, you will have similar success or failure performing a backup with `dump` as you will with `tar` or `cpio`.

How Do I Read This Volume?

If you're a system administrator for long enough, someone eventually will hand you a volume and ask "Can you read this?" She doesn't know what the format is, or where the volume came from, but she wants you to read it. Or you may have a very old backup volume that you wish you could read but can't. How do you handle this? How do you figure out what format a volume is? How do you read a volume that was written on a different machine? These are all questions answered in this section. There are about 10 factors to consider when trying to read an unknown or foreign volume, half of which have to do with the hardware itself—whether or not it is compatible. The other half have to do with the format of the data. If you are having trouble reading a volume, it could be caused by one or more of these problems.

Prepare in Advance

If you've just been handed a volume and need to read it right now, ignore this paragraph. If you work in a heterogeneous environment and might be reading volumes on different types of platforms, read it carefully now. Reading a volume on a platform other than that on which it was created is always difficult.

In fact, except for circumstances like a bad backup drive or data corruption, the only sure way to read a volume easily every time is to read it on the machine that made it. Do not assume that you will be able to read a volume on another system because the volume is the same size, because the operating system is the same, or even if the utility goes by the same name. In fact, don't assume anything.

If it is likely that you are eventually going to have to read a volume on another type of system or another type of drive, see if it works before you actually need to do it. Also, if you can keep one or two of the old systems and drives around, you will have something to use if the new system doesn't work. (I know of companies that have 10- or 15-year-old computers sitting around for just this purpose.) If you test things up front, you might find out that you need to use a special option to make a backup that can be read on other platforms. You may find that it doesn't work at all. Of course, finding that out now is a lot better than finding it out two years from now when you really, really, really need that volume!

Wrong Media Type

Many media types look similar but really are not. DLT, LTO, AIT, and other drives all have different generations of media that work in different generations of drives. If the volume is a tape, and its drive has a media recognition system (MRS), it may even spit the tape back out if it is the wrong type. Sometimes MRS is not enabled or not present, so you assume that the tape should work because it fits in the drive. Certain types of media are made to work in certain types of drives, and if you've got the wrong media type for the drive that you are using, the drive will not be able to read it. Sometimes this is not initially obvious because the drive reports media errors.

Problems involving incompatible media types sometimes can be corrected by using the newest drive that you have available. That is because many newer drives are able to read older tapes created with previous generations of drives. However, this is not always the case and can cause problems.

Bad or Dirty Drive or Tape

If the drive types and media types are the same but one drive cannot read the other drive's tapes, then the drive could be defective or just dirty. Try a cleaning tape, if one is available. If that does not work, the drive could be defective. It also is possible that the drive that wrote the tape was defective. A drive with misaligned heads, for example, may write a backup image that can't be read by a good drive. For this reason, when you are making a backup volume that is going to be stored for a long time, you should verify right away that it can be read in another drive.

Try, Try Again

During a restore, I came across a tape that kept saying it was blank. Of course, the info on the tape was needed for someone who accidentally deleted an important file. After several tries in all four tape drive units in the jukebox, it finally was able to be recognized and read. The restore was done without any further complications. After a lot of breath holding, prayer, and profanity, I realized the moral was, if you at first don't succeed, try, try again!

Ed Lam

Although less common, there are also tape cleaning machines. The machines look like tape drives. They load the tape and run the entire tape through a clean and vacuum process. Sometimes when a tape is unreadable in any drive, cleaning the tape like this can allow the tape to be read. It would be handy to have one of these machines to prepare for such a scenario.

Different Drive Types

This is related to the media-types problem. Not all drives that look alike are alike. For example, not all tapes are labeled with the type of drive they should go into. Not all drives that use hardware compression are labeled as such, either. The only way to know for sure is to check the model numbers of the two different drives. If they are different manufacturers, you may have to consult their web pages or even call them to make sure that the two drive types are compatible.

Wrong Compression Setting/Type

Usually, drives of the same type use the same kind of compression. However, some value-added resellers (VARs) sell drives that have been enhanced with a proprietary compression algorithm. They can get more compression with their algorithm, thus allowing the drive to write faster and store more. If all of your drives are from the same manufacturer, this may not be a problem—as long as the vendor stays in business! But if all your drives aren't from the same manufacturer, you should consider using an alternate compression setting if they have one, such as IDRC or DCLZ. Again, this goes back to proper planning.

The Little Endian That Couldn't

Differences exist among machines of different architectures that may make moving volumes between them impossible. These differences include whether the machine is big-endian, little-endian, ones complement, or twos complement. For example, Intel-based machines are little-endian, and RISC-based machines are big-endian. Moving volumes between these two types of platforms may be impossible.

Most big Unix machines are big-endian, but Intel x86 machines and older Digital machines are little-endian (see Table 23-1). That means that if you are trying to read a backup that was written on an NCR 3b2 (a big-endian machine), and you are using a backup drive on an NCR Intel SVr4 (little-endian) box, you may have a problem. There is also the issue of ones-complement and twos-complement machines, which are also different architectures. It is beyond the scope of this book to explain what is meant by big-endian, little-endian, ones complement, and twos complement. The purpose of this section is merely to point out that such differences exist and that if you have a volume written on one platform and are trying to read it on another, you may be running into this problem. Usually, the only way to solve it is to read the volume on its original platform.

Table 23-1. Big- and little-endian platforms

Big-endian	Little-endian
SGI/MIPS, IBM/RS6000, HP/PA-RISC, Sparc/RISC, PowerPC, DG Aviion, HP/Apollo (400, DN3xxx, DN4xxx), NCR 3B2, TI 1500, Pre-Intel Macintosh, Alpha[2]	DECStations,[3] VAX, Intel x86

[2] I have heard that Alpha machines can actually be switched between big- and little-endian, but I can't find anyone to verify that. But Digital Unix is written for a big-endian alpha, so yours will probably be big-endian.

[3] These are the older DEC 3x00 and 5x00 series machines that run Ultrix.

Most backup formats use an "endian-independent" format, which means that their header and data can be read on any machine that supports that format. Usually, `tar` and `cpio` can do this, especially if you use the GNU versions. I have read GNU `tar` volumes on an Intel Unix or Linux (i.e., little-endian) box that were written on HPs and Suns (i.e., big-endian machines). For example, it is quite common to `ftp` `tar` files from a Unix machine to a Windows machine, then

use WinZip to read them. Again, your mileage may vary, and it helps if you test it out first.

Some people talk about reading a volume with `dd` and using its `conv=swab` feature to swap the byte order of a volume. This may make the header readable but may make the data itself worthless. This is because of different byte sizes (8 bits versus 16 bits) and other things that are beyond the scope of this book. Again, the only way to make sure that this is not preventing you from reading a volume is to make sure that you are reading the volume on the same architecture on which it was written.

Block Size (Tape Volumes Only)

Tape volumes are written in different block sizes, and you often need to know the block size of a tape before you can read it. This section describes how block sizes work, as well as how to determine your block size.

When a program reads or writes data to or from a device or memory, it is referred to as an *I/O operation* . How much data is transferred during that I/O operation is referred to as a *block*. Since the actual creation of each block consumes resources, a larger block usually results in faster I/O operations (i.e., faster backups). When an I/O operation writes data to a disk, the block size that was used for that operation does not affect how the data is physically recorded on the disk; it affects only the performance of the operation. However, when an I/O operation writes to a tape drive, each block of data becomes a tape block, and each tape block is separated by an interrecord gap. This relationship is illustrated in Figure 23-1.

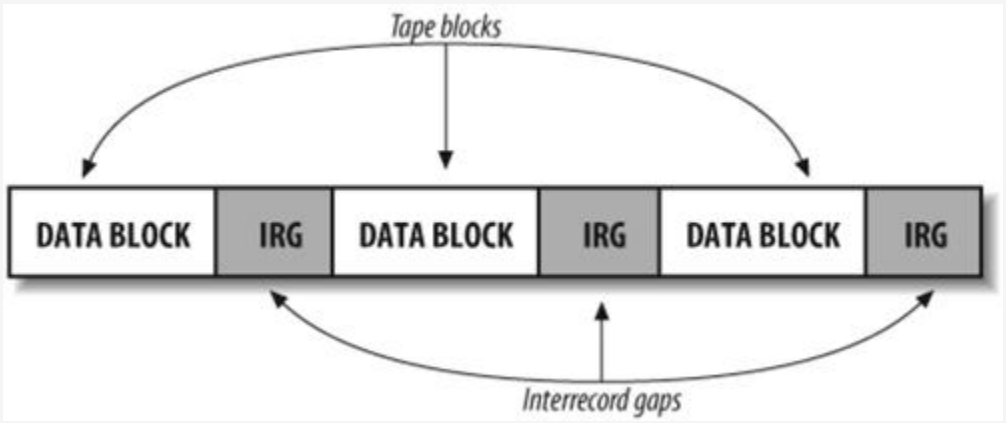


Figure 23-1. Tape blocks and interrecord gaps

All I/O operations that attempt to read from this tape must understand its block size, or they will be unsuccessful. If you use a different block size, three potential scenarios can occur:

Block size is a multiple of the original block size

For example, a tape was recorded with a block size of 1,024, and you are reading it with a block size of 2,048. This scenario is actually quite common and works just fine. Depending on a number of factors, the resulting read of the tape may be faster or slower than it would have been if it used the original block size. (Using a block size that is too large can actually slow down I/O operations.)

Block size is larger than the original block size (but not a multiple)

For example, a tape was recorded with a block size of 1,024, and you are reading it with a block size of 1,500. What happens here depends on your application, but most applications will return an I/O error. The read operation attempts to read a whole block of data, and when it reaches the end of the block that you told it to read, it does not find an interrecord gap. Most applications will complain and exit.

Block size is smaller than the original block size

For example, a tape was recorded with a block size of 1,024, and you are reading it with a block size of 512. This will almost always result in an I/O error. Again, the application attempts to read a block of 512 bytes, then looks for the interrecord gap. If it doesn't see it, it complains and exits.

Interrecord gaps actually take up space on the tape. If you use a block size that is too small, you will fill up a lot of your tape with these interrecord gaps, and the tape actually will hold less data.

Each tape drive on each server has an optimal block size that allows it to stream best. Your job is to find which block size gives you the best performance. A block size that is too small decreases performance; a block size that is too large may decrease performance as well because the system may be paging or swapping to create that large block size. Some operating systems and platforms also limit the maximum block size.

Determine the Blocking Factor

Use the trick described in Chapter 3 in the section "Using dd to Determine the Block Size of a Tape" to determine your block size. If you're reading a `tar` or `dump` backup, you'll need to determine the blocking factor. If the backup utility is `tar`, the blocking factor usually is multiplied by 512. `dump`'s blocking factor usually is multiplied by 1,024. Read the manpage for the command that you are using and determine the multiplier that it uses. Then, divide the block size by that multiplier. You now have your blocking factor.

For example, you read the tape with `dd`, and it says the block size is 32,768. The manpage for `dump` tells you that the blocking factor is multiplied times 1,024. If you divide 32,768 by 1,024, you will get a blocking factor of 32. You

then can use this blocking factor with `restore` to read the tape.

AIX and Its 512-Byte Block Size

Some operating systems, such as AIX, allow you to hardcode the block size of a tape device. This means that no matter what block size you set with a backup utility, the device will always write using the hardcoded block size. During normal operations, most people set the block size to 0, allowing the device to write in any block size that you specify with your backup utility. (This is also known as *variable block size*.) However, during certain operations, AIX automatically sets the block size to 512. This normally happens when performing a `mksysb` or `sysback` backup, and the reason this happens is that a block size of 512 makes the `mksysb`/`sysback` tape look like a disk. That way, the system can boot off the tape because it effectively looks like the root disk. Most `mksysb`/`sysback` scripts set the block size back to when they are done, but not all do so. You should check to make sure that your scripts do, to prevent you from unintentionally writing other tapes using this block size.

Why can’t you read, on other systems, tapes that were written on AIX (with a block size of 512)? The reason is that AIX doesn’t actually use a block size of 512. What AIX *really* does is write a block of 512 bytes and then pad it with 512 bytes of nulls. That means that they’re really writing a block size of 1024, and half of each block is being thrown away! Only the AIX tape drives understand this, which means that a tape written with a block size of 512 can be read only on another AIX system.

However, if you set the device’s hardcoded block size to 0, you should have no problem on other systems—assuming

the backup format is compatible. Setting it to 0 makes it work like every other tape drive. The block size you set with the backup utility is the block size the tape drive writes in. (If you want to check your AIX tape drive's block size now, start up `smit` and choose Devices, then Tape Drives, then Change Characteristics, and make sure that the block size of all your tape drives is set to 0!)

You can even set the block size of a device to 1,024 without causing a compatibility problem. Doing so will force the device to write using a block size of 1,024, regardless of what block size you specify with your backup utility. However, this is a "normal" block, unlike the unique type of block created by the 512-byte block size. Assuming that the backup format is compatible, you should be able to read such a tape on another platform. (I know of no reason why you would *want* to set the block size to 1,024, though.)

To set the block size of a device back to 0, run the following command:

```
# chdev -l device_name -a block_size=0
```

Unknown Backup Format

Obviously, when you are handed a foreign volume, you have no idea what backup utility was used to make that volume. If this happens, start by finding out the block size; it will come in handy when trying to read an unknown format. Then, use that block size to try and read the volume using the various backup formats, such as `tar`, `cpio`, `dump`, and `pax`. I would try them in that order; foreign volumes are most likely going to be in `tar` format because it is the most interchangeable format.

One trick to finding the type of backup format is to take a block of data off of the volume and run the `file` command on it. This often will come back and say `cpio` or `tar`. If that

happens, great! For example, if you used the block size-guessing command shown previously, you would have a file called */tmp/sizefile* that you could use to determine the block size of the tape. If you haven't made this file, do so now, then enter this command:

```
# file /tmp/sizefile
```

If it just says "data," you're out of luck. But you just might get lucky, especially if you download from the Internet a robust *magic* file:

```
# file -f /etc/robust.magic /tmp/sizefile
```

In this case, `file` helps reveal the format for commands and utilities not native to the immediate platform.

Different Backup Format

Sometimes, two commands sound the same but really aren't. This can be as simple as incompatible versions of `cpio`, or at the worst, completely incompatible versions of `dump`. Format inconsistencies between `tar` and `cpio` usually can be overcome by the GNU versions because they automatically detect what format they are reading. However, if you are using an incompatible version of `dump` (such as `xfsdump` from IRIX), you are out of luck! You will need a system of that type to read the volume. Again, your mileage may vary. Make sure you test it up front.

They Used What Kind of Compression?

One day we needed to restore from some older tapes and were having trouble reading them. The drives kept complaining about I/O errors every time we tried to read one of these tapes. After further research, we found out that the tapes had been made on a particular brand of tape drive using their proprietary compression algorithm.

> Unfortunately, this company no longer made the drive. Luckily, we were able to find some refurbished drives that could read the tapes. The first thing we did was to copy them to a tape drive that used a standard compression algorithm.
>
> *Mike Geringer*

Damaged Volume

One of the most common questions I see on Usenet is, "I accidentally typed `tar cvf` when I meant to type `tar xvf`. Is there any way to read what's left on this volume?" The quick answer is no. Why is that?

Each time a backup is written to a tape, an end-of-media (EOM) mark is made at the end of the backup. This mark tells the tape drive software, "There is no more data after this mark—no need to go any further." No matter what utility you try, it will always stop at the EOM mark because it thinks this is the last backup on the tape. Of course, the tape could just be damaged or corrupted. One of the tricks I've seen used in this scenario is to use `cat` to read the corrupted tape:

```
# cat device/tmp/somefile
```

This just blindly reads in the data into */tmp/somefile*, so you can read it with `tar`, `cpio`, or `dump`.

Reading a "Flaky" Tape

One of the fun things about being a backup specialist is that everyone tells you their favorite backup and recovery horror stories. One day a friend told me that he was having a really hard time reading a particularly flaky tape. The system would read just so far into the tape and then quit

with an I/O error. However, if he tried reading that same section of tape again, it would work! He really needed the data on this particular tape, so he refused to give up. He wrote a shell script that would read the tape until it got an error. Then it would rewind the tape, fast-forward (`fsr`) to where he got the error, and try again. This script ran for two or three days before he finally got what he needed. I had never heard of such dedication. I told my friend Jim Donnellan that he had to let me put the shell script in the book. The shell script in Example 23-1 was called *read-tape.sh* and actually did the job. Maybe this script will come in handy for someone else.

Example 23-1. The read-tape.sh script

```
# !/bin/sh

DEVICE=/dev/rmt/0cbn
# Set this to a non-rewinding tape device

touch rawfile
# The rawfile might already be there, but just in case
while true ; do

 size=\Qls -l rawfile | awk '{print $5}'\Q # Speaks for itself

 blocks=\Qexpr "$size" / 512\Q

 full=\Qdf -k . | grep <host> | awk '{print $6}'\Q
 # Unfortunately, this only gets checked once per glitch. Maybe a
fork?

 echo $size
 # Just so I know how it's going

 echo $blocks

 echo $full

 if [ $full -gt 90 ] ; then
      echo "filesystem is filling up"
      exit 1
 fi
```

```
   mt -f $DEVICE rewind
   # Let's not take chances. Start at the beginning.

   sleep 60
   # The drive hates this tape as it is. Give it a rest.

   mt -f $DEVICE fsr $blocks

   # However big rawfile is already, we can skip that on the tape

   dd if=$DEVICE bs=512 >> rawfile # Let's get as much as we can

   if [ $? -eq 0 ] ; then
    # If dd got clipped by a tape error, there's still work to do,
    echo "dd exited cleanly"

    # if not, it must have gotten to the end of the file this time
    # without a hitch. We're done.
    exit 0
   fi

  done
```

If you've got tips on how to read corrupted or damaged volumes, I want to hear them. If I use them in later editions of the book, I will credit your work! (I also will put any new ones I receive on the web site for everyone to use immediately.)

Multiple Partitions on a Tape

This one is more of a gotcha than anything else. Always remember that when a backup is sent to tape, it could have more than one partition on that tape. If you are reading an unknown tape, you might try issuing the following commands:

```
# mt -t device rewind
#mt -t device fsf 1
```

Then, try again to read this backup. If it fails with I/O error, there are no more backups. (That's the EOM marker again.) If it doesn't fail, try the same commands that you

tried in the beginning of the tape to read it. Do not assume that it is the same format as the first partition on the tape. Also understand that every time you issue a command to try and read the tape, you need to rewind it and fast-forward it again using the two preceding commands.

If at First You Don’t Succeed...

Then perhaps failure is your style! That doesn’t mean that you have to stop trying to read that volume. Remember that the early bird gets the worm, but the second mouse gets the cheese. The next time you’re stuck with a volume you can’t read, remember my friend Jim and his flaky tape.

Gigabit Ethernet

As the amount of data that needed to be backed up grew exponentially, backup software became more and more efficient. Advanced features like dynamic parallelism and software compression made backing up such large amounts of data possible. However, the amount of data on a single server became so large that it could not be backed up over a normal LAN connection. Even if the LAN were really fast, only so many bits can be sent over such a wire.

Gigabit Ethernet was supposed to save the backup world. Ten times faster than its closest cousin (Fast Ethernet), surely it would solve the bandwidth problem. Many people, including me, designed large backup systems with gigabit Ethernet in mind. Unfortunately, we were often disappointed. While a gigabit Ethernet connection could support 1,000 Mbps between switches, maintaining such a speed between a backup client and backup server was impossible. The number of interrupts required to support gigabit Ethernet consumed all available resources on the servers involved. Even after all available CPU and memory had been exhausted, the best you could hope for was 500 Mbps. While transferring data at this speed, the systems could do nothing else. This meant that under normal conditions, the best you would get was around 200–400 Mbps.

As of this writing, 10 Gbps NICs are becoming generally available, and *they're* going to solve the world's problems—at a cost of several thousand dollars per NIC. Believe it or not, I've talked to at least one person that was able to achieve 4,000–5,000 Mbps using such a NIC on a high-end Solaris server and Solaris 10's IP stack. If those tests hold true in other shops, it will help. In my heart, however, I think we're fighting a losing battle.

Disk Recovery Companies

It seems fitting that a section in this book should be dedicated to disk recovery companies. When all else fails, these are the guys who might be able to help you. Every once in a while, a disk drive that doesn’t have a backup dies. A disk recovery company actually disassembles this drive to recover its data. This service can cost several thousand dollars, and you pay the fee regardless of the success of the operation. Although these companies may be expensive, and they may not get all the data back, they may be the only way to recover your data. There are several such companies, and they can be found by a web search for “disk recovery.”

Here’s hoping that you never need to use them….

Yesterday

When this little parody of a Paul McCartney song started getting passed around the Internet, it got sent to me about a hundred times! (The original author is unknown.) What better place to put it than here?

Yesterday,

All those backups seemed a waste of pay.

Now my database has gone away.

Oh I believe in yesterday.

Suddenly,

There’s not half the files there used to be,

And there’s a milestone hanging over me

The system crashed so suddenly.

I pushed something wrong

What it was I could not say.

Now all my data’s gone

and I long for yesterday-ay-ay-ay.

Yesterday,

The need for backups seemed so far away.

I knew my data was all here to stay,

Now I believe in yesterday.

Trust Me About the Backups

Here’s a little more backup humor that has been passed around the Internet a few times. This is another parody, attributed to Charles Meigh, based on the song “Use Sunscreen,” by Mary Schmich, which was a rewrite of a speech attributed to Kurt Vonnegut. (He never actually wrote or gave the speech.) Oh, never mind. Just read it!

Back up your hard drive.

If I could offer you only one tip for the future, backing up would be it.

The necessity of regular backups is shown by the fact that your hard drive has a MTBF printed on it, whereas the rest of my advice has no basis more reliable than my own meandering experience.

I will dispense this advice now.

Enjoy the freedom and innocence of your newbieness.

Oh, never mind. You will not understand the freedom and innocence of newbieness until they have been overtaken by weary cynicism.

But trust me, in three months, you’ll look back on groups.google.com at posts you wrote and recall in a way you can’t grasp now how much possibility lay before you and how witty you really were.

You are not as bitter as you imagine.

Write one thing every day that is on topic.

Chat.

Don’t be trollish in other people’s newsgroups.

Don’t put up with people who are trollish in yours.

Update your virus software.

Sometimes you’re ahead, sometimes you’re behind.

The race is long and, in the end, it’s only with yourself.

Remember the praise you receive.

Forget the flames.

If you succeed in doing this, tell me how.

Get a good monitor.

Be kind to your eyesight.

You’ll miss it when it’s gone.

Maybe you’ll lurk, maybe you won’t.

Maybe you’ll meet F2F, maybe you won’t.

Whatever you do, don’t congratulate yourself too much, or berate yourself either.

Your choices are half chance.

So are everybody else’s.

Enjoy your Internet access.

Use it every way you can.

Don’t be afraid of it or of what other people think of it.

It’s a privilege, not a right.

Read the readme.txt, even if you don’t follow it.

Do not read Unix manpages.

They will only make you feel stupid.

Get to know your fellow newsgroup posters.

You never know when they’ll be gone for good.

Understand that friends come and go, but with a precious few, you should hold on.

Post in r.a.sf.w.r-j, but leave before it makes you hard.

Post in a.f.e, but leave before it makes you soft.

Browse.

Accept certain inalienable truths: spam will rise. Newsgroups will flamewar. You too will become an oldbie.

And when you do, you'll fantasize that when you were a newbie, spam was rare, newsgroups were harmonious, and people read the FAQs.

Read the FAQs.

Be careful whose advice you buy, but be patient with those that supply it.

Advice is a form of nostalgia.

Dispensing it is a way of fishing the past from the logs, reformatting it, and recycling it for more than it's worth.

But trust me on the backups.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Chapter 24. It’s All About Data Protection

The chapters that have preceded talk about nothing but backup and recovery, but it’s important to give you a little context before I close out the book. While backup and recovery are extremely important, they are but one part of the data protection landscape. *Data protection* is defined as all activities involved in protecting data from the various things that could happen to it. Data protection activities include backup, recovery, archive, storage security, and disaster recovery, and every company should be aware of all of these activities regardless of its size. While smaller companies may be less subject to some aspects of data protection, such as long-term archiving or compliance issues, they should regularly evaluate each aspect of data protection to determine how much it applies to them—and what they’re doing about it. This is why this book, which focuses on backup and recovery, ends with a chapter on data protection—to make sure that you know that backup and recovery is just the beginning of your job.

Perhaps your company’s archive and storage security requirements are easy to meet, allowing you to fulfill the rest of your company’s data protection requirements with an open-source backup system and a simple off-site tape rotation scheme. Perhaps you’ve got sophisticated archive or storage security needs, and you’re considering implementing an open-source backup and recovery system to leave room in the budget for some of the commercial solutions that address those other elements of data protection. Finally, it’s also possible that you hadn’t considered archiving or storage security prior to reading this chapter. Whichever description best fits you, I hope this overview of data protection gives you some food for thought.

We live in the information age. Even the smallest “mom-and-pop” businesses rely on some type of computer systems to store the information they generate. Perhaps it’s a list of customer phone numbers, a log of business transactions, or even details about a new product. Whatever the information is, the business would be damaged if it was lost, deleted, destroyed, misplaced, or stolen. Therefore, a complete data protection system protects against all these risks.

A data protection system doesn’t just restore data that has been destroyed or damaged. It also helps retrieve data that has been moved or retrieves data that is being requested in a manner different from the way it was stored. It prevents data from falling into the wrong hands and ensures companies are in compliance with regulations affecting their industries. And, of course, a complete data protection system ensures that the data can be restored in times of disaster.

There are both business and technical reasons for data protection. The next two sections provide an overview of those reasons.

Business Reasons for Data Protection

Like many other IT functions, the objective of any data protection strategy is to mitigate risks, reduce costs, or improve service levels.

Mitigating Risk

The primary job of a data protection system is to mitigate risk. In the IT world, risk mitigation is generally synonymous with data availability, internal/external security, and regulatory compliance.

Data availability

Many businesses today require that users and business applications have access to critical information 24 hours a day, 7 days a week, 365 days a year (24×7×365). Businesses that cannot access critical information may be unable to perform one or more key functions such as taking new orders or processing existing claims. Along these same lines, partners of these businesses can experience problems taking and processing orders if they don't have access to this information. Planned and unplanned outages of even a single system can therefore have serious ramifications that affect the business at hand, as well as the businesses of partners and customers.

The consequences of not being able to access data when needed can be serious and can include lost revenue (see the section "Reducing Costs," later in this chapter). Worse still, public news of these types of events can have far-reaching consequences on all parties involved, affecting brand names and reputations.

Internal/external security

Chances are good that the information that drives one business is coveted by another. In fact, to the surprise of many businesses, large and small companies often go to extreme measures, including engaging in corporate espionage, simple mischief, and electronic terrorism, to get access to key competitor customer lists, development plans, and intellectual property.

Identity theft, or identity fraud to some, is another source of concern for IT departments as they carry out data protection strategies. A customer's name or other identifying information (such as address, birth date, and identification numbers like U.S. Social Security numbers) is

about all someone needs to fraudulently empty bank accounts or gain access to credit.

Theft of or access to strategic business information can also have a number of serious business consequences, including loss of competitive advantage, loss of good corporate image, government-imposed fines, and even the loss of a business. All of these have happened to one or more major businesses in the last few years:

- Sadly, more than one company ceased to exist on September 11, 2001 when it was discovered that their hot-site was in the second tower.
- The dreaded “adverse inference”[1] instructions were given in more than one major U.S. lawsuit, resulting in a judgment for the plaintiff and hundreds of millions of dollars in fines.
- The reputation of several major companies was irreversibly damaged when it was revealed that they had not maintained control of the personal information of their customers.

Regulatory compliance

Regulatory compliance adds another layer of IT risk. Businesses today must contend with an increasing number of government and nongovernment regulations. For example, organizations that store the medical information of U.S. citizens are subject to the Health Insurance Portability and Accountability Act (HIPAA). Financial organizations doing business in the U.S. must address the Securities and Exchange Commission (SEC) 17a-4 rule requirements, and industries of all types are accountable to Sarbanes-Oxley (SOX) stipulations. Anyone doing business with residents of the European Union will be very familiar

with the Data Protection Directive of 1998, which governs the control and access of personal information, such as “an identification number or...one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.” Each major country in the world either has or is developing similar regulations. The risks of noncompliance are serious and can include fines that are often in the hundreds of millions of U.S. dollars, prosecution of key corporate officers, or a loss of business such that the organization is forced to close its doors.

Reducing Costs

When data is not protected properly, businesses can rack up a lengthy list of *hard costs* (such as fines levied on an organization in an electronic discovery suit) and *soft costs* (such as missed business opportunities or damaged reputation). An effective data protection strategy is able to minimize these costs by ensuring that data is available to authorized users who need it, when they need it, and according to business objectives.

Downtime costs

If the information that a company uses to generate revenue is unavailable, revenue is lost. However, just how much revenue a company ultimately loses depends on a number of factors including the type of business, the type of data that is unavailable, and how long the data is unavailable. The monetary cost can range from hundreds to millions of U.S. dollars per hour of downtime.

But downtime can also spell missed opportunity, which, though more difficult to quantify, may have equally lasting consequences for revenue. For example, customers who are unable to interact with a company because key information

(such as data needed to process orders) is unavailable may choose to take their business elsewhere either temporarily or permanently. A long-time customer may be more willing to excuse downtime than a first-time customer.

Downtime also translates into a variety of other costs, including wages (to wit, paying employees to do a job they can't perform when the data is unavailable) and even additional storage costs in the form of extra backup copies. Employees, in particular those who do not believe their information is being protected properly, may keep extra copies of critical information on disk, tape, and other storage media. Depending on the amount of extra copies made and the type of storage media used, the added cost can be substantial.

Electronic discovery

Electronic discovery is a term used to refer to the practice of requesting information that has been digitally stored. For example, in a lawsuit a company may be ordered to provide all emails to and from a given employee or containing a certain set of keywords. The SEC may order a brokerage firm to forward all emails containing the words "promise" and "guarantee." The EU may request proof of compliance to the Data Protection Directive. If a company is not set up to retrieve information in this fashion (that is, if this type of information is not immediately available), the costs of satisfying an electronic discovery request can quickly become quite large. While a defendant can petition the court to declare such costs unreasonable, a given judge may or may not grant their petition. Plenty of precedents in various state and federal cases show such requests being denied.

Security breaches

A security breach is the result of a type of unauthorized access to company information. As with downtime, the associated costs of a security breach can be difficult to quantify and can vary greatly among companies. Laws governing security breaches vary by location. Some U.S. states require companies to publicly disclose security breaches that can be financially devastating; others do not. Depending on the industry, a company may or may not be subject to a fine due to a breach. Companies doing business with persons who live in the European Union are subject to how the member country met the “judicial remedies, liability, and sanctions” chapter of the Data Protection Directive.

Data classification

Not all information has the same value within an organization, and it shouldn’t be treated as if it does. Doing so can significantly increase data protection costs. For example, depending on the industry, an archived copy of a three-year-old order may not have the same value as data pertaining to a new product or service. However, many organizations mistakenly apply the same level of data protection to both types of information and store both on costly primary (or high-end) disk.

The key here is to classify information based on the age, regulatory status, and business importance of the data today and over time and then match storage systems and data protection levels accordingly. Companies that follow this type of plan can significantly reduce the total cost of ownership (TCO) of their storage resources.

Improving Service Levels

What does data protection have to do with business service levels? More specifically, how can better data protection help companies improve these levels? It boils down to the age-old dilemma of accessibility versus security: the more available information is to various applications, the greater the potential business benefit. However, the more users who have access to information, the greater the likelihood that someone without proper authorization can gain access to that information.

Companies that “lock down” mission-critical information are missing out on clear opportunities to increase productivity throughout the enterprise. The challenge is finding a way to strike a balance between accessibility and security.

---

[1] Adverse inference is when a judge instructs a jury that the absence of a given piece of proof suggests that the claims the plaintiff is making are correct, even if the actual evidence is not present. It is an extreme measure used only when the judge feels the actual evidence was destroyed or covered up.

Technical Reasons for Data Protection

As previously discussed, data protection is all about keeping company information safe (from accidental or intentional deletion, corruption, and mishandling), available (to authorized users as well as business departments and outside partners), and compliant (with various industry guidelines and governmental regulations), and that doing this is often quite challenging. From a technical standpoint, the job is equally challenging. Disk drive failures, worn tape cartridges, lost or stolen media, and the inherent security risks of network storage all complicate the IT task.

Device Issues

Many of the technical reasons for data protection stem from the characteristics of the many different devices that store data. Every new device increases the chances of failure or attack.

Disk failures

Physical disks (or disk volumes) can fail for a variety of reasons. While the average mean time between failure supplied by the vendor of an expensive disk drive is quite high, other factors such as usage, handling, and environmental conditions can affect the reliability of disk drives. Disk failures can also result from outside factors, such as fires, natural disasters, and acts of physical terrorism.

Tape media wear, stolen/misplaced tapes

Data is written to tape in a sequential fashion rather than in a random fashion like disk; consequently, both the tape drive and tape media can experience significant wear and tear during frequent read and write actions. Tape media is also highly susceptible to environmental factors such as humidity and heat.

Because data is typically written to tape in an unencrypted plain-text format (such as `tar2`), unauthorized users can rather easily retrieve information from the media. Therefore, stolen or misplaced tapes can result in a significant exposure to a company's intellectual property and the personal information of its customers. This is why it's important to have good physical security of your backup media.

Networked storage risks

Implementing storage networks, whether storage area networks (SANs) or network-attached storage (NAS) environments, is a two-edged sword in terms of pros and cons. On the one hand, storage networks can significantly improve data availability and manageability. On the other hand, they can open new security risks.

Prior to the advent of networked storage, ATA or SCSI disk drives were directly attached to host servers in a local area network (LAN) environment. (This is now referred to as direct-attached storage, or DAS.) With DAS, the only way to compromise the data on these drives was to compromise the security of each individual host. Because servers were "insulated" from widespread hacking, companies were able to set up different security levels in the LAN depending on the datatype.

In a networked storage environment, the situation is different. It is possible, depending on the configuration and

the security level of a given storage network, to access multiple hosts' data from a compromised host without physically hacking into each host. If one host is compromised and is able to "see" the other hosts' disks, the hacker can gain access to the data on those hosts without physically compromising those servers, too. Access to communication between these servers is enabled over network protocols, such as Unix-based NFS, Windows-based Common Internet File System (CIFS), Fibre Channel, or IP-based iSCSI.

A hacker, if so motivated, could also attack hosts, as well as stage other attacks, from the vantage point of the storage. The hacker could do things within the storage environment that would wreak havoc in the computing environment, such as denying all hosts access to shared data. Suddenly all your data is unavailable, and you have no idea why.

Email Can Be Critical

I was the email administrator for a medical software company, and I warned our system administrators that we needed to back up certain email stores more often and to put the email stores on a RAID-protected disk. Both of them blew me off. About a month after I became the email admin, the CEO, CIO, CFO, and "owner" email stores were corrupted due to disk issues (the physical disk was failing). It took some real creative work on all of our parts to get the data back so we didn't lose the docs on a multimillion dollar deal. (We were a 50-person company, and this was a huge deal for us.) After that day, I put in a mirror drive and hosted the CXO's and owner's email stores. The backup admin started reading the daily reports on what successfully backed up and what didn't. (He had been getting a daily report that was being sent automatically to his trash.)

Scott Boss

External Threats

In addition to risks introduced by the types of devices that are in use, there are threats introduced by the people who use or have access to the devices. These include viruses, worms, Trojan horses, and, of course, accidental or intentional deletion.

Viruses

Microsoft's definition of a virus is a good place to start. It defines a virus as "a piece of computer code that attaches itself to a program or file so it can spread from computer to computer, infecting [software, hardware, and files] as it travels."

Worms

A worm, like a virus, is designed to copy itself from one computer to another, but, unlike viruses, it does so automatically by taking control of features on the computer that can transport files or information. Once you have a worm in your system, it travels alone; it does not have to attach itself to a program or file to wreak havoc with your system or others.

Worms are also dangerous because they can replicate in great volume through email address books. The result can be devastating, causing heavy network traffic that can bring business networks to their knees and slow down Internet traffic considerably.

Trojan horses

As for Trojan horses, just as the mythological Trojan horse tricked the city of Troy into believing it was receiving a gift, today's Trojan horses, which often come in the form of email attachments, trick users into believing they are receiving security updates or other important information. In reality, they are recipients of hidden viruses that attempt to disable antivirus and firewall software.

Whether the threat is a virus, worm, or Trojan horse, it is extremely important for companies to protect themselves against these types of attacks. Once they're in, they can be extremely difficult to purge from your environment.

Accidental deletion

Whether companies admit it or not, data loss caused by human error is a common occurrence. It takes the form of accidental deletion of a single file, an entire filesystem, or a user from a network configuration. The only way to recover from this type of internal threat is through a historical copy such as a backup or snapshot.

Intentional deletion

Some data is deleted because a malevolent person decides it should be deleted. There was a major incident a few years ago when a malicious employee deleted every file on every server and desktop at a major financial institution—all because he didn't get the raise he wanted. A good data protection system might notice that this has happened or even prevent it from happening. At a minimum, a good data protection system should be able to restore the data in question.

Backup and Archive

The first two elements of data protection are backup and archive. These are related but very different activities. Backup is copying data from one place to another in case the original is damaged. Archive is copying or moving data to long-term storage for quick retrieval of logical components for a specific business purpose. The following comparison between backup and archive clarifies the important distinction between these two activities:

Backups are the secondary copy of primary data.

The purpose of backups is to recover primary data if it’s damaged, deleted, or corrupted. Therefore, a backup is a secondary copy of primary data.

Archives are the primary copy of secondary data.

In contrast, an archive is created to be the primary copy of data that is important, but not important enough to be placed on primary storage. This includes old files that are being moved to secondary storage to save space as well as second copies of primary data that are created for a secondary purpose, such as content searching.

Backups recover data that was damaged, deleted, or corrupted.

This is the only purpose for backups. If a file is deleted or damaged, you can restore it from backup copies. Note that in order to perform a restore, you usually need to know the server and filesystem the file resided in and possess the application that created it.

Archives retrieve data from secondary storage.

While the words restore, recover, and retrieve can be used as synonyms in certain contexts, they have different

meanings in the context of backup and archive. Restore and recover refer to putting something back in its original condition whereas retrieval refers to obtaining it from an alternate storage location. This is why we *restore* or *recover* from backups, but we *retrieve* from archives, as in “I retrieved the folder from the file drawer.”

Archives retrieve data in a manner other than that in which it was stored.

Many applications create numerous completely disconnected pieces of data that are stored in files, emails, and databases. Very few applications are able to search across information stored in multiple applications, filetypes, and files stored over different points in time, but modern archiving applications are meant to do just that. A dynamic archiving system, such as an email system, can monitor and archive all incoming and outgoing email and allow you to search for content across multiple emails, email servers, and points in time. Some filesystem products can do the same for files. In other words, archives can retrieve data in a manner other than which it was stored.

Backups are stored only long enough to cover the usage pattern of the data.

Backups need to be stored only long enough to be able to recover deleted files. How long they need to be stored is based on the usage pattern of the data. If some files are only accessed once a quarter, you should be keeping backups longer than a quarter. If a file is only accessed once a quarter, and you keep backups for only one month, you cannot restore a file that was deleted three months ago and not accessed until yesterday.

Archives can be stored for many years or decades.

Sometimes archives contain information that’s been deleted from primary storage and archived in case the

information is ever needed again, such as the design plans for a product a company no longer makes. Someone needs to decide when that information can be deleted from the archives. Sometimes archives contain information governed by regulations, and those regulations dictate when that information can be deleted. The result is that archives can be stored for years or even decades.

Document Your Changes

At a large oil company, prior to a company holiday, a backup operator decided to change the next full to an incremental because there would not be a large rate of data change. The admin promptly forgot about the change. Three months later when a restore was needed, it was discovered that they needed to pull from three months of incrementals to do the restore.

David Bregman

What Needs to Be Backed Up?

Just about everything in a corporate environment needs to be backed up. The more important question to ask is what needs to be restored and how quickly does it need to be restored. As discussed in the disaster recovery section, you must define recovery requirements for every piece of data in your company. The primary reason for performing backups is to provide continuous access to a corporation's data in the event the primary copy of data is unavailable. Companies need to back up three types of data:

Intellectual property

: This is the information about a company's core competency. In the case of a biotech firm, it is access to data captured in a discovery process; for a market research firm, it is access to database records.

Customer data

: Examples range from scanned copies of patient x-rays to market research information and records about the buying patterns of particular market segments. It can also include information that can be used to conduct identity theft, such as a customer's address, birth date, or identification number.

Operational data

: This last category includes every other kind of data in the organization. It can include data about where organizations purchase supplies to build products to information about who is responsible for the delivery of products to customers. It includes payroll and accounting information and any other type of information that isn't intellectual property or the personal information of customers.

What Needs to Be Archived?

Beyond backing up data, organizations must also develop a strategy for archiving data. Both are integral components of an effective data protection strategy and necessitate a clear understanding of the business value of data (archiving maybe even more so than backup).

When executed correctly, archiving not only can save organizations money but it can also be a lifesaver, especially for those requiring access to historical information for regulatory compliance or audit purposes. Conversely, when archiving is performed incorrectly, it can cost a company dearly in terms of lost revenue, fines, and other penalties.

The problem is that many organizations mistakenly think of backup as archiving, and vice versa. The confusion regarding archiving often comes from some backup vendors that claim that their products also have archiving capabilities. Frequently, these capabilities equate to nothing more than backing up a data set and then deleting that dataset from primary storage. This is not archiving.

Vendor products offer different levels of “archiving” capabilities. At one end of the spectrum, some vendors treat archiving as simply a backup followed by a deletion of the data from primary storage. This type of “archiving” is really intended to assist organizations in removing old data that is cluttering up servers—a problem that is better addressed with storage resource management (SRM) or hierarchical storage management (HSM) tools.

So, what is archiving and how does it fit into the data protection landscape? Archiving is the long-term storage of information for the retrieval of logical components for a specific business purpose. In comparison, backups are

intended to protect against short-term data loss, such as accidental deletion, device failure, and data corruption.

Archival data candidates include periodic corporate financial information that needs to be retained for auditing purposes, medical patient information that must be retained for HIPAA compliance purposes, and clinical trial data for a new drug that is winding through the Food and Drug Administration drug approval process. Other examples include email, check images, and other types of electronic communication that could be requested in an audit.

The long-term nature of archived data presents a number of new problems:

Backward compatibility

: Because tape and optical drives typically can’t read media that is more than a generation or two old, organizations must give some thought to the long-term recovery of data that is archived to tape. Data can be migrated to new tape or optical platforms, but migration can present validation and authentication issues in some regulated industries.

Media longevity

: If data is to be maintained for a long time on tape or optical media, steps must be taken to ensure media integrity. This includes maintaining proper environmental control and refreshing volumes as needed.

Readability/usability of the data after a restore

: The archived data must be “portable.” The archived data cannot depend on an obsolete version of an application or operating platform in order to be restored.

For both archiving and backup, it is critical to develop an understanding of the corporate value of the data to be protected. Deciding what data should be archived, when it should be archived, and how long it should be stored is central to the storage management process. A system of data classification like this can lead to intelligent policy management of both primary (disk) and secondary (backup and archive) storage resources.

Examples of Backup and Archive

The following two examples illustrate the differences between information that is archived and information that is backed up. Keeping backup copies for several years does not inherently make them archive copies. They are simply backups that have been kept for an extended period of time.

WingsRUs, a fictitious aircraft manufacturer, has detailed information about its latest plane, the WingsRUs 563. The data includes critical marketing information, detailed design specifications, invoices for materials, testing plans and results, FAA inspection and approval information, and customer information. The data is stored on different databases on different servers at its various locations. WingsRUs backs up this data regularly using traditional backup applications.

When the company begins manufacturing the next new plane, the WingsRUs 565, it determines it no longer needs to back up data regarding the 563; instead it needs to make room for the new 565 data, which is now the more valuable data to the company.

In the event that there are issues with the plane, the company decides to retain much of the 563 data in case the FAA should ask for detailed design specifications, testing plans, or other related material in the future. WingsRUs migrates its 563 backup data to a reference archive system, which provides appropriate access to this information when needed at a price point that is cost-effective. If the data is properly archived, the company should be able to retrieve the old plans by simply asking for the 563 without needing to know (or have access to) the hostnames, filesystems, or applications that stored the data in the first place.

A U.S. financial trading firm communicates with its customers primarily via email. After receiving a series of complaints that this firm is reportedly “promising” a specific rate of return on certain investments, the Securities and Exchange Commission begins an investigation. The investigation begins with an *electronic discovery request* to see all email written in the last two years that contain the words “promise” and “guarantee.” A discovery request is a legal term for a request for background information on a particular subject. An electronic discovery request is a discovery request that requires you to obtain the information from your computer systems.

Without an email archiving system, this type of request is difficult or even impossible to meet. If the firm had performed daily backups of its email system for the last two years, it would need to restore 730 versions (365 backup copies×2 years) of its Exchange Server and then search each version for the requested words—at best a daunting task. However, if the emails had been archived to an email archiving system, the company could easily search all emails sent in the last two years for the words “promise” and “guarantee,” and they would be immediately presented with a copy of all such messages.

Can Open-Source Backup Do the Job?

Electronic information is stored in a number of different ways, some of which require special treatment during the backup process. Ignoring these differences can have a number of negative side effects, including:

- A significant decrease in backup performance
- An even larger decrease in restore performance
- An inability to recover the data or system in question

How information is treated depends on how the information is stored. The standard, or most common, way that information is stored is as a file in a filesystem. The most common example of a filesystem is the “C:\” drive on Windows desktops and the “/” on Unix-based systems. Most backup products handle ordinary static files without issue, but some filesystems and datatypes can cause problems and often need to be treated specially. They include:

- Very active filesystems
- Filesystems with large (more than 1 TB) volumes of data
- Filesystems with millions files
- Information stored inside databases
- Metadata not stored in a filesystem or database
- Link file structures and device files
- Information stored on NAS-based filesystems
- Information stored on SAN-based filesystems

Most commercial backup products have additional features to handle these datatypes, usually at an additional price. This section considers whether the open-source products covered in this book can handle the same challenges.

Very Active Filesystems

The basic assumption of traditional backup software is that a file is not changing while it is being backed up. In the past, IT managers put systems into single-user mode prior to performing the backup to ensure that files were static, or unchanging, during the backup process. IT managers often do not have the luxury of doing this today, so backup application vendors have developed techniques for backing up these types of files.

Constantly changing files present a special challenge to backup and recovery software applications—even commercial products. In addition, some operating systems and applications can lock files for exclusive use, preventing even backup applications from accessing them. If a file is too active or locked during backup, commercial backup systems use snapshot techniques to ensure that the files are protected. Snapshot technologies present a static view of the filesystem to the backup application. If this particular challenge is present in your Windows environment, you may want to investigate how your open-source product integrates with Windows snapshot services. If this challenge is present in your Unix or Macintosh environment, you may find that you're not able to solve it without moving to a commercial product.

Very Large Filesystems

Very large filesystems present another set of unique backup challenges. Because traditional backup software applications are filesystem-based, each filesystem or drive is backed up separately. While this method works fine with small to moderate-sized filesystems that are dozens or hundreds of gigabytes in size, it does not perform well with filesystems that are greater than 1 TB in size.

The problem is that the speed of the fastest tape drives can push data at a rate of about 200 MB/s (at this writing). If you could supply a stream of data fast enough, you could back up a 1 TB filesystem in approximately two hours. The problem is that you probably couldn't keep up with the 200 MB/s tape drive, and it would take significantly longer than that.

If this particular challenge is present in your world, you may consider near-continuous data protection. A near-continuous data protection backup system uses replication techniques to maintain a copy of the data for backup purposes. Since it never has to do a full backup, it needs a lot fewer resources to run successfully. Since it's disk-based, it's also going to be able to back up and recover as fast (or as slow) as the filesystem you're trying to back up. See Chapter 7 for a discussion of three open-source near-CDP products.

Filesystems with Too Many Files

Filesystems with lots of files also present a set of unique challenges to IT managers. In fact, a 1 GB filesystem with five million files is actually as challenging to back up as a 1 TB system with a few thousand files. Why? Because of the number of operations that must be performed during the backup and restore process.

The problem is even worse on the recovery side, which requires even more steps to complete. For example, the backup application must first tell the operating system that it needs to create a file; it then needs to open that file for writing and transfer the data into that file. After the restore, a series of checks is performed to verify that the data written to the filesystem is the same as the data that was backed up.

Again, each of these operations takes time, and because they are performed in sequence, they can actually bring the restore process to a screeching halt. The speed of the backup device is irrelevant. The fastest disk drive or tape drive in the world is still going to have to sit and wait while each of these operations is performed for each file.

An alternative approach to traditional backup and restore processes is *image-based backup*, which bypasses the filesystem (and files) and backs up data at the block level. Unfortunately, any open-source project designed to address this challenge probably requires the drive to be unmounted during backup. If this limitation is unacceptable, you have to switch to a commercial product.

Information Stored in Databases

Information stored in databases can be difficult to back up because of the way it is stored, the changing nature of the datafiles, and demanding recovery point objectives, or RPOs.

Databases generally store databases in files in the filesystem, but some databases store “raw” data, or datafiles, directly on disk. While storing data on raw disk can improve performance, it can make the backup environment significantly more complex.

Datafiles change constantly; therefore, the challenge is to create a consistent image of the datafile during the backup process. A variety of techniques, including cold backups, scripted hot backups, and database backup agents, can help IT managers create these images.

A *cold backup* is the backup of datafiles after a database has been shut down. A *scripted hot backup* places the database in some type of backup mode before backing up its datafiles using a regular backup program. Either method works well with most of the backup utilities covered in this book. A *database backup agent* interfaces directly with the database for backup purposes. At this writing, none of the open-source backup products support backing up any database using its agent. However, many of the database agents do support backing up to disk without a commercial backup product. For example, Oracle's `rman`, Sybase and SQL Server's `dump database`, and `ntbackup`'s Exchange plug-in all support backing up to disk while the database is active. You could therefore create a disk-based backup that is then backed up by the open-source backup system you chose.

Databases generally have more demanding RPOs. While it may be acceptable to restore a word processing file to last night's backup, it is generally not acceptable to restore a database file from a backup copy that is several hours old. Databases provide *transaction logs* that track changes in between backups. A proper backup of a production database includes a system for backing up the transaction log during the day.

Information Stored on Shared Storage

Information stored on shared storage can also create extra backup requirements. Shared storage comes in two main flavors: SAN and NAS. SANs are based on the SCSI

protocol and allow several systems to have block access to shared disk or tape drives. SANs typically run either SCSI over Fibre Channel or IP (iSCSI). NAS is based on NFS or CIFS protocols, which allow multiple servers to share files across an IP network.

SAN-based filesystems

The low-cost backup products covered in this book are not going to treat SAN-based filesystems any differently than a locally attached filesystem. If you need a product that performs SAN-based backups, you need a commercial product.

NAS-based filesystems

Although the snapshot and off-site replication software offered by some NAS vendors has great recovery features, NAS filers must still be backed up at some point. All of the open-source products discussed in this book are going to back up NAS-based filesystems via a share (NFS or CIFS).

Disaster Recovery

Devising a good disaster recovery (DR) plan is a bit like how many cities paint their bridges. The painters start at one end of the bridge and paint until they reach the end. Then they go to the other side of the bridge and start painting it until they reach the other end. Then they start all over again; they basically never stop painting the bridge.

So should it be when you're building your DR plan. You have to build it from the ground up, and it can take months or even years to perfect, at which point you have to go back to the beginning and start all over. Since computer environments are changing constantly, you continually have to change and test your plan to make sure it still works.

This section is not meant to be a comprehensive guide to disaster recovery planning. There are books dedicated to just that topic, and before you attempt to design your own disaster recovery plan, I strongly advise you to research this topic further. This part of the chapter gives you an overview of the steps necessary to complete such a plan as well as a few details that are typically left out of other books.

Everything Starts with the Business

The design of any disaster recovery system should be driven by the ability to make available to the business the critical systems and information systems required to conduct normal production activities, without making those systems and information available to the wrong people. We are not protecting this data because it is a school project or an interesting hobby. We are protecting the data because if the data is lost, the ability to conduct business operations is at risk. Thus, it all starts with the business.

Define the Core Competency of the Organization

When looking at data protection, the first question to ask is, “What are the core products and/or services that this organization offers?” followed by “What is the information required to provide that product or service, and what applications are required to effectively use the information?” The answer to the second question is what defines your organization’s intellectual property (IP). Without these information systems, the organization would not be able to function.

Many types of information qualify as IP. For example, it could be your version of KFC’s 11 herbs and spices—the “secret recipe” that makes your company’s product or service different from everyone else’s product or service. Of course, without customers that secret sauce is not worth much. All of your customers’ locations, names, and contact information are also part of your IP, as are the names of potential customers. Any plans that your company has for doing something different, reaching a different market, or

selling to a new group of people are also part of your IP. If it is information that you do not want in the hands of your competitor, it is part of your IP. This broad definition can include many types of information.

Intellectual property is a wonderful thing, but it is not the only important information in your company. Circling around the creation and delivery of your product or service are a number of other systems, such as procurement, payroll, accounts receivable and accounts payable, sales, and customer support. Each system is also critical to your business, and each needs a particular set of information to perform its function.

Prioritize the Business Functions Necessary to Continue the Core Competency

Once you identify all intellectual property and supporting applications and systems, you must prioritize the business functions necessary to continue providing your company's core products or services. This phase is not just about importance; you must also consider urgency or criticality as well.

In order to establish what IP needs to be protected, you must understand the organization's core competencies. Next, you need to prioritize the protection and recovery of these systems should they become unavailable. If the core competency relies on the manufacturing of a product, the systems to continue the process are vital to the continuation of the business. Other systems, such as email, may not be vital to the core competency and do not require the same level of protection. However, systems supporting customer communications may be critical and thus perhaps these email systems should be treated as critical applications as well. It is important to understand what is

vital and critical to the organizations supporting the business's core competency and not just protect whatever data happens to reside on your servers.

Correlate Each System to a Business Function, and Prioritize

Let us consider a power company as an example. If it did not deposit customer payments for a few weeks, some people might notice and many would not care, but the company's creditors would notice and care. Some overly conscientious customers might notice that their checks had not cleared yet, but it would not bother most of them. The company's creditors would only notice if it failed to make payment on a payable account. Even then, the company could probably explain to its creditors that it is in the middle of some sort of emergency, and the creditor would probably hold off the firing squad. However, what would happen if it stopped delivering electricity or gas for just a few minutes? It would be on the evening news, all its business customers would be angry at the impact to them, all the residential customers would have to reprogram their DVD players and microwaves, and the company could potentially cause a rolling blackout, similar to what happened in the U.S. Northeast in the early 2000s. (This happens in some parts of the world on a regular basis.) This means that the company's ability to deliver power is the most critical business function it has—its core competency.

Once you figure out what your IP and supporting systems are, and which ones are critical, you need to figure out where they reside and all of the resources required to use them. Is the information stored in a database? Is it stored in files on a filesystem somewhere? In most cases, the data

is going to be stored on some type of computer system. Every computer and storage system must be assigned to a business function based on that business unit's level of criticality, thereby giving that system the same recovery priority as the business function to which it belongs.

A great example of this type of prioritization can be found in a publication of the U.S. Federal Communications Commission. It shows the FCC's different types of data and its criticality, and it is published at http://www.fcc.gov/webinventory/. Interestingly enough, its most critical systems are those required by law or presidential decree. It lists mission critical as the next level of criticality, followed by frequently requested data, and other data. For reference purposes, most companies use the term "mission critical" to describe their most important systems. In this case, the FCC has acknowledged that without governmental decree, it would have no mission. Therefore, it has another level higher than mission critical. The important thing to learn here is that each industry and company is different, and you must perform this prioritization of business functions specifically for your organization.

Define RPO and RTO for Each Critical System

Your recovery time objective, or RTO, is how quickly you want the system to be recovered. RTOs can range from zero seconds to many days, or even weeks. Each application serves a business function, so the question is how long you can live without that function. If the answer is that you cannot live without it for one second, then you have an RTO of zero seconds. If the answer is that you can

live without it for two weeks, you have an RTO of two weeks.

The recovery point objective, or RPO, defines the point in time that is reflected once you have recovered a system, also referred to as how much data you can afford to lose. Consider two examples: customer orders and system logs. If you lose one customer order, the company loses significant revenue Therefore, many companies determine that they cannot lose any customer orders. That means they have an RPO of zero seconds for customer orders. On the other hand, system logs might be useful only when troubleshooting problems or when auditing systems. If you lose several days of them, it is a problem only if you need to troubleshoot a problem or audit the logs from that time period. However, if you acknowledge that the time period is lost anyway (due to a disaster), it is more important to just get the order system running immediately; the logging system is not as time-sensitive as the order system. Therefore, you can lose one week of system logs without really losing anything critical to the business. That means there is a one-week RPO for system logs.

Once you have established a priority for each system and determined the various outages that you are going to protect against, you must create an RTO and RPO for each system that is to be protected. Most of your customers really do not care what causes an outage or delay, so the RTO and RPO should be the same in all but the most extreme events, like a catastrophic earthquake affecting an entire region. Depending on their level of criticality, most systems have the same RTO and RPO for each disaster type. For some systems, however, you may find that a longer RPO and RTO is acceptable or unavoidable for major disasters.

Create Consistency Groups

It is often necessary to recover several systems to the same point in time. This is primarily caused by applications that pass data to one another. Consider a manufacturing company with the business processes of sales and custom manufacturing. There are possibly several different computer systems involved in this process, including the customer, orders, procurement, and manufacturing databases. If this business has a customer expecting a product, hopefully all four systems know that. What would happen, for example, if it was manufacturing a custom product and lost the original order, or it had the order, but didn't know which customer it went to? What would happen if it took the order, but the manufacturing database became corrupted, and the company did not know it was supposed to be making a product? This would represent a serious integrity problem.

Therefore, if your company has several systems that perform related business processes, those systems need to be in the same consistency group. In addition to determining an RTO and RPO, you must identify those systems that are related to each other because they need to be recovered to the same point in time. It is also important to identify a consistency window, or a window of time during which not all affected systems are changing.

If your consistency window is larger than (or the same as) your backup window, it's relatively easy to meet the consistency requirement for a consistency group. For example, a 5 p.m. to 8 a.m. window is a 15-hour consistency window. If all systems are down between 5:00 p.m. and 8:00 a.m. (no new data is being created), and backups start and finish sometime between 5 p.m. and 8 a.m., it does not matter if System A is backed up at 10:00

p.m. and System B is backed up at 2:00 a.m. They will still be in sync.

However, if your consistency window is too short to back up all systems in a consistency group, you need to do one of two things. One option is to create a custom backup window for those systems and ensure that they back up within that window. This option is certainly preferable because it does not significantly complicate your backup system. If your consistency window is too short for this approach, the second option is to augment your backup system with snapshots or business continuance volumes (BCVs). They allow you to quickly create a “virtual backup” on disk of several systems within a few seconds and then later convert the virtual backup to a physical backup (that is, back up the snapshot to tape or virtual tape).

Determine for Each Critical System What to Protect from

Once the business functions are prioritized, and each system is assigned to a business function, it is time to identify the things that can happen that trigger a recovery scenario. “Disasters” come in many forms. First, create a list of the different levels and types of disasters that are likely for your area and type of business.

> **Tip ✎**
>
> The Disaster Recovery Institute states that each company should define its own levels of disasters. I’ve listed the way I define them, which starts with a loss of a single system.

Level 1 disasters are those that take out an entire application or server:

- Disk or disk array outage
- Internal corporate sabotage
- Electronic terrorism (denial-of-service attack)
- Disgruntled employee attack

Level 2 disasters are those that take out an entire data center:

- Building fire or flood
- Natural disasters (hurricane, tornado, earthquake)
- Building condemnation (chlorine gas leak)
- Physical terrorism (bomb)
- *Really* disgruntled employee
- Loss of all network connectivity
- Loss of all electrical power

Level 3 disasters take out an entire campus, city, or metropolitan area:

- Large-scale natural disasters
- Physical terrorism (bombing of power plant)
- Act of war

Determine the Costs of an Outage

Once you have determined all of the different types of disasters and their associated probability, you must assign a cost to each type of disaster, for each type of system. For

example, if a fire took out your test server for a week, your cost may be nothing. However, if a fire took out a server that you deemed in the previous exercise to be mission critical, a loss of only a few minutes may cost you millions, depending on the level of criticality and the business you are in.

Such costs can come from a number of areas, starting with the loss of business. While your systems are down, you are not taking orders, making your product, or delivering your service. Another cost is a loss of reputation, which can result in the loss of future business. No company wants to be on CNN because it lost its data. Labor costs must also be added to the equation, and there are two kinds of labor costs. The first labor cost is when data was created and then lost; work has to be redone. The second type of labor cost is the opportunity cost of labor for workers who are not doing anything useful because the system is down. Depending on the type of business you are in, there may also be the loss of source materials used to create your product. For example, if you are a food maker of some sort, and the automation control systems are down, your ingredients may expire before the product is completed.

Another concept to think about when calculating the cost of an outage is that outage costs are logarithmic. Some costs can be avoided if the outage is minimal. A five-minute outage, for example, can be overlooked as a nice break for your employees in the middle of a busy workday. However, as the outage gets longer and longer, other contingency plans go into effect and the costs of the outage start increasing. Before you know it, companies that count on you for your product start looking elsewhere, and then things get really out of control.

Plan for All Types of Disasters

Many companies attempt to perform risk assessments of all possible disaster situations, determining for each the likelihood that it will happen to any particular data center. For example, coastal regions usually prepare for hurricanes or tsunamis. In the U.S., parts of Texas, Oklahoma, Kansas, and Nebraska have had so many tornadoes that they call it “Tornado Alley.” Other parts of the world are more susceptible to earthquakes. Do not dismiss any particular type of disaster with a “that will never happen” type statement. Murphy’s Law will find you. The disaster you do not prepare for is the one that will strike you.

Whereas natural disaster or malicious actions seem to be the most obvious cause of outage, accidental causes are probably most common. While people frequently accidentally delete files or misplace them, complete data loss has been caused by power outages when construction workers in the area cut power lines to the data center. Other common occurrences include water damage from broken water pipes and software upgrades gone awry. You should undertake some research to consider all of the possible types of disasters and make sure your disaster recovery plans take each of them into account.

Prepare for Cost Justification

Once you begin the process of selecting data protection systems, you need to justify the cost of each purchase. To be successful in doing so, you must have completed the steps mentioned previously in this section:

1. Define your RTOs, RPOs, backup windows, and consistency group requirements.
2. Determine for each critical system what to protect.
3. Determine the cost of each outage.

4. Plan for all types of disasters.

Once you have accomplished this, justifying the cost of each data protection system should be a relatively easy thing to do. You simply need to state your required RTOs and RPOs, what you're protecting against, and what a system to protect against those things costs. If any part of the system is turned down, you simply need to explain how that affects your ability to meet these requirements.

Storage Security

Although the details of storage security are really beyond the scope of this book, it is important to understand that security is a very important part of data protection. This section gives you an overview of the vulnerabilities in storage systems.

Plain-Text Communication

In a storage network, we refer to communications within the network, such as a host requesting data from a storage device, as *in-band*. Historically, all of this communication has been in plain text. If someone can view in-band traffic, she might be able to read data she's not supposed to, or to learn something that might assist in an attack. We refer to communications outside the network—perhaps someone managing a storage device via its IP management port—as *out-of-band*. If someone can view out-of-band management information, they could take control over the storage network and give themselves access to information, or conduct denial-of-service (DOS) attacks.

The key to solving both of these problems is encryption. For out-of-band communication, more and more storage vendors are supporting secure communication protocols, such as ssh or https, on their management ports. For in-band support, there are host-based encryption systems and hardware encryption appliances. Only host-based encryption can encrypt data from the point of departure, but encryption software has typically been very CPU-intensive, slowing down the transfer of data by as much as 50 percent. The other in-band choice is an encryption appliance that can go in the storage network and encrypt

data as it’s stored on the device, preventing readability even if a hacker is able to gain physical access.

Poor Authentication and Authorization Systems

Unix’s NFS and Windows’ CIFS allow the sharing of files between multiple servers. This is collectively referred to as network-attached storage, or NAS. A major challenge with NFS and CIFS is their simple host-based authentication mechanisms. If your IP address resolves to the appropriate hostname, you are given access to the shared directory. In addition, much of the authentication mechanisms are also sent in plain text, telling a hacker exactly what addresses he needs to spoof. A hacker could easily spoof the appropriate IP address and be given access to the wrong information.

Fibre Channel SANs have authentication and authorization issues as well. Two very insecure, but very common, practices are the use of World Wide Name-based (WWN-based) zoning and soft zoning. (A *zone* is the Fibre Channel equivalent of a VLAN, with some differences.) Let’s first take a look at the authentication issue, then we’ll look at the authorization issue.

The authentication issue with Fibre Channel is the common use of WWN-based zones, where zone membership is determined by a host’s WWN, which is equivalent to a MAC address. The problem with using WWNs for authentication is that they are easily spoofed. The ability to change the WWN is built right into the driver.

A much more secure, albeit slightly harder to manage, authentication method would be to specify zone membership using the switch port a given host is plugged into. Port binding, a recent advancement in Fibre Channel

switches, can also improve WWN-based authentication. Using this authentication method, a WWN is bound to a particular port and is granted access only if it is seen at that port.

There are also authorization problems in Fibre Channel SANs, especially when using *soft zoning*. With soft zoning, you won't be able to query the name server to get members of a zone if you're not in that zone, but you can still communicate with a device if you have its WWN, which is relatively easy to determine. The opposite of soft zoning is *hard zoning*. With hard zoning, only members of a zone can access the devices in that zone.

> **Tip ✎**
>
> Many people believe that soft zoning and WWN-based zoning are the same, and that hard zoning and port zoning are the same. This comes from the long-standing practice of offering them together. Nonetheless, they are two very different concepts. WWN-based and port-based zoning specify zone membership. Hard zoning and soft zoning specify whether or not zone membership is required to communicate with a member of a zone.

While the solution of using only hard zoning seems simple, it hasn't been that easy. Historically, soft zoning went hand in hand with WWN-based authentication, and many people use WWN-based authentication to make changes more easily. Today's switches are beginning to let you independently choose which authentication and zoning methods you want to use. The most secure combination, of course, would be hard zoning with port-binding-based authentication.

Backup Flaws

Backup systems' most obvious security flaw is the plain-text backup tape. There are many new encryption options for protecting this media. They include host-based filesystem and application encryption, encryption in the backup software, and a number of appliances that sit in the hardware data path and encrypt the data as it is written to tape. (Some of these appliances are now available inside the tape library or tape drive.) These hardware appliances are the most expensive, but they are much easier to implement and maintain than the other options. In addition to encrypting at line speed and providing superior key management, they also support compression. Since encrypted data can't be compressed, some have a compression chip that compresses the data before it's encrypted. This gives these appliances a major advantage over the other solutions, such as application encryption and backup encryption, because their encrypted data is not compressed by the tape drive.

The next security issue with backup systems is that they have typically used hostname-based authentication to authenticate the backup server and client to each other. A hacker with a spoofed IP address could do two things to exploit this vulnerability. First, she could create a rogue backup client and ask the server to restore data for the real client, thus stealing the information. A rogue client could also populate the backup server with bogus versions of backed-up files. A malicious hacker could also create a rogue backup server and back up any client that the server is authorized to back up. This, of course, would be a perfect way to steal or corrupt all kinds of data. Some backup products, including some of the open-source products discussed in this book, have addressed this serious vulnerability with additional levels of authentication

beyond the hostname. Unfortunately, the added complexity of such authentication systems has made them less than attractive to backup administrators.

Most backup systems have taken an “all or nothing” approach to administrative authorization. This means that someone can do everything or nothing at all within the backup system. For example, by giving a new administrator the ability to eject tapes from the library, you also give them the ability to delete or change every backup policy, delete all backup history, and overwrite every tape you own with garbage. This presents the possibility of a novice administrator pushing the wrong button, accidentally erasing all the tapes in your tape library. A healthcare company actually had that happen a few years ago. Some backup software products have begun resolving this problem by introducing role-based administration, so you can give each person only the capabilities he needs to do his job.

The introduction of role-based administration in backup software, along with other new functionality to secure stored data, shows that storage vendors are waking up to the importance of security. If your products don’t support this kind of secure functionality, put pressure on your vendors so they understand it’s critical for the safety of your most precious data.

Conclusion

Companies of all sizes must ensure that they are taking care of all parts of the data protection landscape that pertain to them. Unfortunately, the open-source projects that have enabled this book to exist have not yet embraced the archive, security, and disaster recovery elements of data protection in the same way they have embraced backup and recovery. While some aspects of archiving, security, and disaster recovery have been addressed in some open-source projects, we don't yet have open-source answers to the commercial products that are addressing today's archiving, security, and disaster recovery needs. Once we have open-source email archiving products, line-speed encryption products, or full-scale block-level replication products, it will be time for another edition of this book. For now, I hope this book has been helpful.

Tip ✎

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at http://www.backupcentral.com.

Index

A note on the digital index ✎

A link in an index entry is displayed as the section title in which that entry appears. Because some sections have multiple index markers, it is not unusual for an entry to have several links to the same section. Clicking on any link will take you directly to the place in the text in which the marker appears.

Symbols

8 mm drives, 8 mm (8x0x) drives (end-of-lifed)

A

access time, Don’t Forget Unix mtime, atime, and ctime
access to backup volumes, limiting, 12,000 gold pieces
ACID compliance, ACID Compliance
administration, Backing up only selected drives or filesystems, Ease of Administration
    ease of, Ease of Administration
    multiple backups, problems with, Backing up only selected drives or filesystems
Advanced Maryland Automated Network Disk Archiver, Amanda (see Amanda)
Advanced Metal Evaporative (AME) media, AIT drive
afio utility, Conclusions
AIX bare-metal recovery, AIX Bare-Metal Recovery, AIX 5.x Operating System, IBM’s mksysb and savevg Utilities, Setting a mksysb Definition for a Client, mksysb and savevg Format, Backing Up to Disk, Making a Bootable DVD/CD

from an Existing mksysb, Setting Up NIM, savevg Operations, Verifying a mksysb or savevg Backup, System Cloning
backing up to disk, Backing Up to Disk
bootable DVD/CD, Making a Bootable DVD/CD from an Existing mksysb
mksysb utility, IBM’s mksysb and savevg Utilities, Setting a mksysb Definition for a Client
Network Install Manager (NIM), setting up, Setting Up NIM
savevg utility, mksysb and savevg Format, savevg Operations
system cloning, System Cloning
verifying mksysb or savevg backup, Verifying a mksysb or savevg Backup
AIX operating system, AIX Bare-Metal Recovery, Backing Up rootvg to Locally Attached Tape, AIX 4.x Operating System, AIX 5.x Operating System, AIX and Its 512-Byte Block Size
4.x operating system, AIX 4.x Operating System
5.x operating system, AIX 5.x Operating System
backup and recovery, AIX Bare-Metal Recovery
block size, hardcoding, AIX and Its 512-Byte Block Size
tape devices, ways to access, Backing Up rootvg to Locally Attached Tape
Amanda, Amanda, Future Plans, Client/Server Architecture Using Nonproprietary Tools, Client/Server Architecture Using Nonproprietary Tools, Client/Server Architecture Using Nonproprietary Tools, Amanda Security, Amanda Security, Amanda Security, Amanda Security, Holding Disk, Backup Scheduling, Tape Management, Device Management, Configuring Amanda, Backing Up Clients via NFS or Samba, Amanda Recovery, Community and Support Options, Community and Support Options
.amandahosts files, Amanda Security

backing up via the Network File System (NFS) or Samba, Backing Up Clients via NFS or Samba
backup scheduling, Backup Scheduling
configuring, Configuring Amanda
device management, Device Management
documentation, Community and Support Options
enterprise support, Community and Support Options
holding disk, Holding Disk
open source and, Client/Server Architecture Using Nonproprietary Tools
OpenSSH, Amanda Security
recovery, Amanda Recovery
scalability, Client/Server Architecture Using Nonproprietary Tools
security, Amanda Security
tape host, Client/Server Architecture Using Nonproprietary Tools
tape management, Tape Management
amrecover, Amanda Recovery
amrestore, Amanda Recovery
asr (Apple System Restore), asr
atime, Don’t Forget Unix mtime, atime, and ctime
atomicity, Atomicity and partial transfers
attributes, Don’t Forget Unix mtime, atime, and ctime, Attribute
files, ctime value and, Don’t Forget Unix mtime, atime, and ctime
autoloader, Automated Backup Hardware
automation, Backing up the entire system, Automate Your Backup, Keep It Simple, SA, Electronic vaulting, Automate Bare-Metal Recovery with G4L
backups, Backing up the entire system, Automate Your Backup, Keep It Simple, SA
bare-metal recovery with G4L, Automate Bare-Metal Recovery with G4L
off-site storage process, Electronic vaulting

B

backing up databases, Backing Up Databases, Unique Database Requirements, Can It Be Done?
    filesystem backups versus, Can It Be Done?
backup and recovery utilities, Basic Backup and Recovery Utilities, Using ssh or rsh as a Conduit Between Systems, An Overview, psync, rsyncx, hfstar, xtar, and hfspax, How Mac OS Filesystems Are Different, asr, pax, psync, rsyncx, hfstar, xtar, and hfspax, Backing Up and Restoring with ntbackup, Using System Restore in Windows, Backing Up with the dump Utility, Using the index to create a table of contents, Restoring with the restore Utility, Specifying no query during restore (y), Backing Up and Restoring with the cpio Utility, Using cpio's Directory Copy Feature, Backing Up and Restoring with the tar Utility, Restoring to an alternate location, Backing Up and Restoring with the dd Utility, Using rsync, Backing Up and Restoring with the ditto Utility, Comparing tar, cpio, and dump
    asr (Apple System Restore), asr
    comparison table, Comparing tar, cpio, and dump
    cpio utility, Backing Up and Restoring with the cpio Utility, Using cpio's Directory Copy Feature
    dd utility, Backing Up and Restoring with the dd Utility
    ditto utility, Backing Up and Restoring with the ditto Utility
    dump utility, Backing Up with the dump Utility, Using the index to create a table of contents
    Mac OS X, how they differ, How Mac OS Filesystems Are Different
    ntbackup, Backing Up and Restoring with ntbackup
    overview, An Overview, psync, rsyncx, hfstar, xtar, and hfspax
    pax (portable archive exchange), pax

psync, rsyncx, hfstar, xtar, and hfspax, psync, rsyncx, hfstar, xtar, and hfspax
restore utility, Restoring with the restore Utility, Specifying no query during restore (y)
rsync, Using rsync
System Restore (Windows), Using System Restore in Windows
tar utility, Backing Up and Restoring with the tar Utility, Restoring to an alternate location
backup device, different drives, problems with, Wrong Media Type
backup hardware, Backup Hardware, Disk-As-Tape: Virtual Tape Cartridges, Decision Factors, Summary, Reliability, Reliability, Duty Cycle, Transfer Speed, Flexibility, Disk drives: Very flexible, Time-to-Data, Capacity, Removability, Cost, Using Backup Hardware, Nearline and Offline Storage, Compression, Density Versus Compression, How Often Should I Change My Media?, Cartridge Care, Drive Care, Nearline and Offline Storage, Tape Drives, VXA, Tape Drives Must Be Streamed, Variable Speed Tape Drives, Helical and Linear Tape Drives Are Different, Cartridges Versus Cassettes, Midrange Tape Drive Types, Optical Drives, UDO recording format, Automated Backup Hardware, Automated Backup Hardware, Automated Backup Hardware, Automated Backup Hardware, Automated Backup Hardware, Automated Backup Hardware, Disk-As-Disk Targets, SAN disk-as-disk targets, NAS disk-as-disk targets, Disk-As-Tape: Virtual Tape Libraries, Packaging, De-duplication, Replication, Content-awareness, Content-awareness, Re-presentation, Stacking, Notification, Disk-As-Tape: Virtual Tape Cartridges, Disk-As-Tape: Virtual Tape Cartridges
autoloader, Automated Backup Hardware
automated, Automated Backup Hardware
capacity, Capacity
cartridge care, Cartridge Care

cartridge versus cassette tape drives, Cartridges Versus Cassettes
compression, Compression
content-awareness, Content-awareness
cost, Cost
de-duplication and, De-duplication
decision factors, Decision Factors, Summary
density versus compression, Density Versus Compression
disk drives versus tape media, Reliability
disk-as-disk targets, Disk-As-Disk Targets
drive care, Drive Care
duty cycle, Duty Cycle
flexibility, Flexibility
helical versus linear tape drives, Helical and Linear Tape Drives Are Different
jukebox, Automated Backup Hardware
library, Automated Backup Hardware
midrange tape drive types, Midrange Tape Drive Types
multiplexing, Disk drives: Very flexible
NAS disk-as-disk targets, NAS disk-as-disk targets
nearline and offline storage, Nearline and Offline Storage
notification, Notification
number of passes, and media life, How Often Should I Change My Media?
optical drives, Optical Drives, UDO recording format
packaging, Packaging
re-presentation, Content-awareness
reliability, Reliability
removability, Removability
replication and, Replication
SAN disk-as-disk targets, SAN disk-as-disk targets
silo, Automated Backup Hardware
stacking, Stacking
streaming tape drives, Tape Drives Must Be Streamed

tape drives, Tape Drives, VXA
time left powered off, Re-presentation
time-to-data, Time-to-Data
transfer speed, Transfer Speed
using, Using Backup Hardware, Nearline and Offline Storage
variable-speed tape drives, Variable Speed Tape Drives
virtual tape cartridges (VTCs), Disk-As-Tape: Virtual Tape Cartridges
virtual tape libraries, Disk-As-Tape: Virtual Tape Cartridges (see VTL)
VTL, Disk-As-Tape: Virtual Tape Libraries
backup servers, Take an Inventory
recovering, Take an Inventory
backup utilities, Torture-Testing Backup Programs, Other warnings
native, Torture-Testing Backup Programs
versions, problems with, Other warnings
backup volumes, Backing up only selected drives or filesystems, Storing Your Backups, 12,000 gold pieces, Following Proper Development Procedures, cpio’s Restore Options, How Do I Read This Volume?, Block Size (Tape Volumes Only), Unknown Backup Format, Different Backup Format, Damaged Volume, Reading a “Flaky” Tape, Multiple Partitions on a Tape
cpio format, cpio’s Restore Options
damaged, reading problems, Damaged Volume
different formats, reading, Different Backup Format
inventorying, 12,000 gold pieces
long-term archives, ensuring readability, Following Proper Development Procedures
multiple backups, dividing among, Backing up only selected drives or filesystems
reading, How Do I Read This Volume?
storing, Storing Your Backups

tapes, Block Size (Tape Volumes Only), Reading a “Flaky” Tape, Multiple Partitions on a Tape
unknown format, Unknown Backup Format
BackupCentral.com, Why the Word “Volume” Instead of “Tape”?
BackupPC, BackupPC Features, The Future of BackupPC, How BackupPC Works, Installation How-To, Installation packages, Configuration Files, Per-Client Configuration, The BackupPC Community, The BackupPC Community
community, The BackupPC Community
configuration files, Configuration Files
installation, Installation How-To, Installation packages
overview, How BackupPC Works
per client configuring, Per-Client Configuration
web site, The BackupPC Community
backups, Don’t Go Overboard, Get the Coverage That You Need, Why the Word “Volume” Instead of “Tape”?, Backing It All Up, The Impossible Job That No One Wants, The Impossible Job That No One Wants, Deciding What to Back Up, Back Up All or Part of the System?, Deciding When to Back Up, Backup Levels, Which Levels Do You Run and When?, “In the Middle of the Night...”, Deciding How to Back Up, Automate Your Backup, Backups change atime, Keep It Simple, SA, Storing Your Backups, Testing Your Backups, Monitoring Your Backups, rsync with Snapshots, Restoring from the Backup, How large is each backup?, Keep Backup Traffic at the Subnet Level, Protection of the Backup Index, Choosing Backup Methods, Four backup options, Four backup options, Four backup options, Four backup options, Four backup options, Step 1: Back Up Important Metadata, Step 2: Back Up the OS with a Native Utility, mksysb and savevg Format, mksysb and savevg Format, mksysb and savevg Format, Preparing to Use mksysb and savevg, Verifying a mksysb or savevg Backup, Preparing for a Bare-Metal Recovery, Backing Up Databases, Unique Database Requirements, Do It Yourself:

Creating Your Own Backup Utility, Calling a Professional, Oracle Backup and Recovery, A Broken Record, Sybase Backup and Recovery, Bring the database online, SQL Server, Master Database Restore, Exchange, Performing a basic restore, Backup, Verifying the Backup, PostgreSQL, Restoring from a Point-in-Time Backup, MySQL, Restoring a MySQL cluster, VMware and Miscellanea, Using Bare-Metal Recovery to Migrate to VMware, Corrupted or Unreadable Backup, Torture-Testing Backup Programs, Torture-Testing Backup Programs, Using Snapshots to Back Up a Volatile Filesystem, Summary of dump steps, Unknown Backup Format, Different Backup Format

atime, changing, Backups change atime
automating, Automate Your Backup
choosing methods, Choosing Backup Methods
commercial database backup products, Calling a Professional
cost, weighing, Get the Coverage That You Need
creating your own database backup utility, Do It Yourself: Creating Your Own Backup Utility
databases, Backing Up Databases, Unique Database Requirements
deciding what data to include, Deciding What to Back Up
distributing backup devices to the subnet level, Keep Backup Traffic at the Subnet Level
dump utility, Summary of dump steps
Exchange, Exchange, Performing a basic restore
Exchange Server, Backup, Verifying the Backup
excluding files, Preparing to Use mksysb and savevg
formats, Unknown Backup Format, Different Backup Format
how to do, deciding, Deciding How to Back Up
importance of, The Impossible Job That No One Wants
iPod, using for, Preparing for a Bare-Metal Recovery
levels, Backup Levels

metadata, Step 1: Back Up Important Metadata
mksysb, mksysb and savevg Format
mksysb utility, mksysb and savevg Format, Verifying a mksysb or savevg Backup
monitoring, Monitoring Your Backups
MySQL, MySQL, Restoring a MySQL cluster
operating system, backing up with a native utility, Step 2: Back Up the OS with a Native Utility
options, Four backup options, Four backup options, Four backup options, Four backup options, Four backup options
    alt-boot filesystem, Four backup options
    alt-boot full image, Four backup options
    alt-boot partition image, Four backup options
    live, Four backup options
Oracle, Oracle Backup and Recovery, A Broken Record
personnel, The Impossible Job That No One Wants
PostgreSQL, PostgreSQL, Restoring from a Point-in-Time Backup
protection of the backup index, Protection of the Backup Index
reasons for, rsync with Snapshots
recoverability requirements, defining, Don’t Go Overboard
restoring from, Restoring from the Backup
savevg, mksysb and savevg Format
scheduling, Deciding When to Back Up, Which Levels Do You Run and When?, “In the Middle of the Night…”
simplicity, retaining, Keep It Simple, SA
size, considering, How large is each backup?
SQL Server, SQL Server, Master Database Restore
storing, Storing Your Backups
Sybase, Sybase Backup and Recovery, Bring the database online
systems, Back Up All or Part of the System?

tape versus volume (CDs or magneto-optical disks), Why the Word “Volume” Instead of “Tape”?
testing, Testing Your Backups
torture-testing, Torture-Testing Backup Programs
VMware, VMware and Miscellanea, Using Bare-Metal Recovery to Migrate to VMware
volatile filesystems, Corrupted or Unreadable Backup, Torture-Testing Backup Programs, Using Snapshots to Back Up a Volatile Filesystem
Bacula, Bacula, Plug-in Support for File Daemons, Bacula, Bacula Architecture, Bacula Components, Authentication, Configuration, Bacula Features, An Example Configuration, Setting Up the Server, Mac OS X Backup, Advanced Features, Bare-Metal Recovery, Bare-Metal Recovery, Backup Traffic and Storage Encryption, Python Script Support, Client Script Support, Autochanger Support, ANSI and IBM Tape Labels, File-Based Intrusion Detection, Pool Migration, Tracking Deleted/Renamed Files, Python-Based GUI Tool, Base Job Support, Client-Initiated Backups, Plug-in Support for File Daemons
ANSI and IBM tape labels, ANSI and IBM Tape Labels
architecture, Bacula Architecture
authentication, Authentication
autochanger support, Autochanger Support
backup traffic and storage encryption, Backup Traffic and Storage Encryption
bare-metal recovery, Bare-Metal Recovery
BartPE rescue CD, Bare-Metal Recovery
base job support, Base Job Support
client script support, Client Script Support
client-initiated backups, Client-Initiated Backups
components, Bacula Components
configuration, Configuration, An Example Configuration
example, An Example Configuration
documentation, Advanced Features
features, Bacula Features

intrusion detection, File-Based Intrusion Detection
Mac OS X backup, Mac OS X Backup
plug-in support for file daemons, Plug-in Support for File Daemons
pool migration, Pool Migration
project homepage, Bacula
Python script support, Python Script Support
Python-based GUI tool, Python-Based GUI Tool
server setup, Setting Up the Server
tracking deleted or renamed files, Tracking Deleted/Renamed Files
bar codes (backup volumes), 12,000 gold pieces, Choosing a media vaulting vendor
bare-metal recovery, Solaris Bare-Metal Recovery, Final Thoughts, Solaris Bare-Metal Recovery, Final Thoughts, Linux and Windows, Commercial Solutions, Live or alternate boot?, Live Method, Automate Bare-Metal Recovery with G4L, Commercial Solutions, HP-UX Bare-Metal Recovery, System Recovery and Disk Mirroring, AIX Bare-Metal Recovery, AIX 5.x Operating System, AIX Bare-Metal Recovery, Mac OS X Bare-Metal Recovery, Recover the System
AIX, AIX Bare-Metal Recovery, AIX 5.x Operating System
automated with G4L, Automate Bare-Metal Recovery with G4L
commercial solutions, Commercial Solutions
HP-UX, HP-UX Bare-Metal Recovery, System Recovery and Disk Mirroring
IBM tools, AIX Bare-Metal Recovery
Linux and Windows, Linux and Windows, Commercial Solutions
live method, Live Method
live versus alternative boot, Live or alternate boot?
Mac OS X, Mac OS X Bare-Metal Recovery, Recover the System

Solaris, Solaris Bare-Metal Recovery, Final Thoughts
big-endian platforms, The Little Endian That Couldn’t
BLOB, Large object datatypes (see large object (LOB) data)
block, Blocks
block size, Using dd to Determine the Block Size of a Tape, mksysb and savevg Format, Block Size (Tape Volumes Only), AIX and Its 512-Byte Block Size
backups, mksysb utility, mksysb and savevg Format
hardcoding on operating systems, AIX and Its 512-Byte Block Size
tape backups, differences in, Block Size (Tape Volumes Only)
tape, determining with the dd utility, Using dd to Determine the Block Size of a Tape
blocking factor, determining (tar), Determine the Blocking Factor
booting from alternative media, How It Works
bosinst.data file, Preparing for the restore
byte-order problems, swapping bytes, incompatibility problems, The Little Endian That Couldn’t

C

capacity, Capacity
cartridge care, Cartridge Care
cartridge versus cassette tape drives, Cartridges Versus Cassettes
cascading, Replication
cat command, reading corrupted tapes with, Damaged Volume
CD recording formats, CD recording formats
CD-ROMs, as backup medium, Why the Word “Volume” Instead of “Tape”?
certificate authority (CA), Backup Traffic and Storage Encryption

champagne backup on a beer budget, Champagne Backup on a Beer Budget
change time, Don’t Forget Unix mtime, atime, and ctime
choosing backup methods, Choosing Backup Methods
client-side compression, Use Client-Side Compression
cloning, system, System Cloning, System Cloning
cold backups, Information Stored in Databases
commercial backup utilities, Commercial Backup Utilities, Final Thoughts, What to Look For, Full Support of Your Platforms, Should You Back Up Special Files?, Backup of Raw Partitions, Backup of Very Large Filesystems and Files, Backup of Very Large Filesystems and Files, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, LAN-Free Backup, LAN-Free Backup, Server-Free (or Serverless) Backup, De-Duplication Backup Systems, Snapshots, Replication, Near-Continuous Data Protection Systems, Continuous Data Protection Systems, Remote Office Backup, Simultaneous Backup of Many Clients to One Drive, Disk-to-Disk-to-Tape Backup, Simultaneous Backup of One Client to Many Drives, Data Requiring Special Treatment, Network-Mounted Filesystems, Custom User Scripts, Databases, Storage Management Features, Information Lifecycle Management, Archives, Satisfy electronic discovery requests, Hierarchical Storage Management, Information Lifecycle Management, Reduction in Network Traffic, Incorporate Throttling, Storage Area Networks, Support of a Standard or Custom Backup Format, Standard Backup Formats, Ease of Administration, Security, Ease of Recovery, Protection of the Backup Index, Robustness, Automation, Volume Verification, Cost, Vendor
    automation, Automation
    cost, Cost

custom backup formats, Support of a Standard or Custom Backup Format
data requiring special treatment, Data Requiring Special Treatment, Network-Mounted Filesystems, Custom User Scripts, Databases
    custom user scripts, Custom User Scripts
    databases, Databases
    network-mounted filesystems, Network-Mounted Filesystems
disk-to-disk-to-tape backup, Disk-to-Disk-to-Tape Backup
ease of administration, Ease of Administration
ease of recovery, Ease of Recovery
network traffic, reducing, Reduction in Network Traffic
platform support, Full Support of Your Platforms
protection of backup index, Protection of the Backup Index
raw partitions, Backup of Raw Partitions
requirements, aggressive, Aggressive Requirements
requirements, extremely aggressive, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, Aggressive Requirements, LAN-Free Backup, LAN-Free Backup, Server-Free (or Serverless) Backup, De-Duplication Backup Systems, Snapshots, Replication, Near-Continuous Data Protection Systems, Continuous Data Protection Systems, Remote Office Backup
    continuous data protection (CDP), Continuous Data Protection Systems
    de-duplication backup systems, De-Duplication Backup Systems
    LAN-free, LAN-Free Backup
    near-CDP, Near-Continuous Data Protection Systems
    recovery point objective (RPO), Aggressive Requirements
    recovery time objective (RTO), Aggressive Requirements

remote office backup, Aggressive Requirements, Remote Office Backup
replication, Replication
server-free (serverless) backup, Server-Free (or Serverless) Backup
snapshots, Snapshots
storage area network (SAN), LAN-Free Backup
very critical applications, Aggressive Requirements
very large applications, Aggressive Requirements
restores, testing, Aggressive Requirements
robustness, Robustness
security, Security
simultaneous backup of many clients to one drive, Simultaneous Backup of Many Clients to One Drive
simultaneous backup of one client to many drives, Simultaneous Backup of One Client to Many Drives
special files, Should You Back Up Special Files?
standard backup formats, Standard Backup Formats
storage area network (SAN), Storage Area Networks
storage management, Storage Management Features, Information Lifecycle Management, Archives, Satisfy electronic discovery requests, Hierarchical Storage Management, Information Lifecycle Management
archives, Archives
electronic discovery requests, Satisfy electronic discovery requests
Hierarchical Storage Management (HSM), Hierarchical Storage Management
Information Lifecycle Management (ILM), Information Lifecycle Management
switching products, Backup of Very Large Filesystems and Files
throttling and, Incorporate Throttling
vendors, Vendor
very large filesystems and files, Backup of Very Large Filesystems and Files

volume verification, Volume Verification
what to look for, What to Look For
compatibility problems (different media types), Wrong Media Type
complete disk versus separate partitions, Complete disk or separate partitions?
compression, Compression
content-awareness, Content-awareness
continuous data protection (CDP), Continuous Data Protection Systems
copy procedure (backup volumes), checking, Testing your chosen vendor
cost, What Will Lost Data Cost You?, What Will Downtime Cost You?, Get the Coverage That You Need, Be ready, Prepare for Cost Justification
backups, Get the Coverage That You Need, Be ready, Prepare for Cost Justification
data losses, determining, What Will Lost Data Cost You?
downtime, determining, What Will Downtime Cost You?
cost of backup hardware, Cost
cpio utility, Backups change atime, Backing Up and Restoring with the cpio Utility, Using cpio’s Directory Copy Feature, The Syntax of cpio When Backing Up, The Options to the cpio Command, The Options to the cpio Command, Restoring with cpio, Using cpio’s Directory Copy Feature, Torture-Testing Backup Programs, Demystifying dump
atime, changing, Backups change atime
Directory Copy feature, Using cpio’s Directory Copy Feature
filesystem-based, Demystifying dump
GNU, The Options to the cpio Command
options, The Options to the cpio Command
restoring with, Restoring with cpio
syntax, The Syntax of cpio When Backing Up
volatile filesystem backups, Torture-Testing Backup Programs

ctime, Don’t Forget Unix mtime, atime, and ctime, The atime can be reset—with a penalty
    backup utilities, changing, The atime can be reset—with a penalty
custom backup formats, Support of a Standard or Custom Backup Format, Custom Backup Formats
customers, losing through data loss, What Will Lost Data Cost You?
customization, image.data or bosinst.data files, Preparing for the restore

D

damaged backup volumes, reading problems, Damaged Volume
data, You Never Want to Say These Words, What Will Lost Data Cost You?, Deciding What to Back Up
    backups, Deciding What to Back Up
    losing, unacceptability of, You Never Want to Say These Words
    losses, cost of, What Will Lost Data Cost You?
data fork, Don’t Forget Mac OS Resource Forks
data protection, It’s All About Data Protection, Conclusion, Business Reasons for Data Protection, Prepare for Cost Justification, Mitigating Risk, Regulatory compliance, Reducing Costs, Improving Service Levels, Device Issues, Networked storage risks, External Threats, Backup and Archive, Examples of Backup and Archive, Can Open-Source Backup Do the Job?, Very Active Filesystems, Very Large Filesystems, Information Stored in Databases, Information Stored in Databases, Information Stored in Databases, Information Stored on Shared Storage, Disaster Recovery, Everything Starts with the Business, Define RPO and RTO for Each Critical System, Define RPO and RTO for Each Critical System, Create Consistency Groups, Create Consistency Groups, Determine the Costs of an Outage,

Prepare for Cost Justification, Storage Security, Poor Authentication and Authorization Systems, Poor Authentication and Authorization Systems, Backup Flaws
backup and archive, Backup and Archive, Examples of Backup and Archive
business continuance volumes (BCVs), Create Consistency Groups
business issues, Business Reasons for Data Protection, Prepare for Cost Justification, Everything Starts with the Business
cold backups, Information Stored in Databases
cost justification, Prepare for Cost Justification
database backup agents, Information Stored in Databases
device issues, Device Issues
disaster recovery (DR), Disaster Recovery
email, Networked storage risks
external threats, External Threats
improving service, Improving Service Levels
mitigating risk, Mitigating Risk
open-source, Can Open-Source Backup Do the Job?
outage costs, Determine the Costs of an Outage
poor authentication and authorization, Poor Authentication and Authorization Systems
recovery point objective (RPO), Define RPO and RTO for Each Critical System
recovery time objective (RTO), Define RPO and RTO for Each Critical System
reducing costs, Reducing Costs
regulatory compliance, Regulatory compliance
scripted hot backups, Information Stored in Databases
shared storage, Information Stored on Shared Storage
soft versus hard zoning, Poor Authentication and Authorization Systems
storage security, Storage Security

synchronization requirements, Create Consistency Groups
systemic flaws, Backup Flaws
very active filesystems, Very Active Filesystems
very large filesystems, Very Large Filesystems
databases, Schadenfreude, Don’t Go Overboard, Don’t Go Overboard, Take an Inventory, Are You Backing Up What You Think You’re Backing Up?, 12,000 gold pieces, Test Everything!, Strategy: Make backups an integral part of the installation process, Backing Up Databases, Unique Database Requirements, Confusion: The Mysteries of Database Architecture, The Muck Stops Here: Databases in Plain English, What Can Happen to an RDBMS?, Database Structure, The Power User’s View: Logical Elements of a Database, Checkpoint, Instance, Database, Table, Table, Index, LOBs, Object, Row, Attribute, The DBA’s View: Physical Elements of a Database Environment, Checkpoint, Page, Datafile, Extents, Extents, Tablespace, Partition, Master database, Master database, Transaction, Rollback log, Transaction log, Checkpoint, An Overview of a Page Change, ACID Compliance, What Can Happen to an RDBMS?, Backing Up an RDBMS, SQL Server, Physical and Logical Backups, Physical and Logical Backups, Physical and Logical Backups, Physical and Logical Backups, Physical and Logical Backups, Get Every Instance, Transaction Log Dumps Are Not Incremental Backups, Do It Yourself: Creating Your Own Backup Utility, Calling a Professional, DB2, Exchange, Informix, MySQL, Oracle, PostgreSQL, Sybase, SQL Server, Restoring an RDBMS, Loss of Any Nondata Disk, Loss of a Data Disk, Loss of a Data Disk, Online Partial Restores, Documentation and Testing
ACID compliance, ACID Compliance
architecture, mysteries of, Confusion: The Mysteries of Database Architecture
attribute, Attribute

backing up, Backing Up an RDBMS, SQL Server
backing up every instance, Get Every Instance
backup volumes, 12,000 gold pieces
backups, Schadenfreude, Don’t Go Overboard, Strategy: Make backups an integral part of the installation process
checkpoint, Checkpoint
cold backup, Physical and Logical Backups
commercial backup products, Calling a Professional, DB2, Exchange, Informix, MySQL, Oracle, PostgreSQL, Sybase, SQL Server
    DB2, DB2
    Exchange, Exchange
    Informix, Informix
    MySQL, MySQL
    Oracle, Oracle
    PostgreSQL, PostgreSQL
    SQL Server, SQL Server
    Sybase, Sybase
control file, Master database
creating your own backup utility, Do It Yourself: Creating Your Own Backup Utility
datafile, Datafile
defined, Database
Exchange versus, Database Structure
extent, Extents
hot backups, Physical and Logical Backups
incremental backups versus transaction log dumps, Transaction Log Dumps Are Not Incremental Backups
index, Index
instances, Instance
instances on server, listing, Are You Backing Up What You Think You’re Backing Up?
large object (LOB) data, LOBs
logical backup, Physical and Logical Backups

logical elements, The Power User's View: Logical Elements of a Database, Checkpoint
master database, Master database
object, Object
page (block), Page
page change, overview, An Overview of a Page Change
partition, Partition
physical backup, Physical and Logical Backups
physical elements, The DBA's View: Physical Elements of a Database Environment, Checkpoint
raw devices versus cooked files, Physical and Logical Backups
RDBMS, overview, The Muck Stops Here: Databases in Plain English, What Can Happen to an RDBMS?
restores, Test Everything!
restoring, Restoring an RDBMS, Loss of Any Nondata Disk, Loss of a Data Disk, Loss of a Data Disk, Online Partial Restores, Documentation and Testing
  documentation and testing, Documentation and Testing
  loss of a data disk, Loss of a Data Disk
  loss of a nondata disk, Loss of Any Nondata Disk
  loss of the master database, Loss of a Data Disk
  online partial restores, Online Partial Restores
rollback log, Rollback log
row (tuple), Row
segments, Tablespace
set up information, recording, Take an Inventory
tables, Table
tablespace, Extents
transaction, Transaction
transaction log, Transaction log
transaction logs, importance of, Don't Go Overboard
view, Table
what can go wrong, What Can Happen to an RDBMS?

DB2, DB2 Architecture, Managing archive logs, Instance, Databases, Schemas, Tables, Views, Indexes, DB2 engine dispatch units, System catalog tables, Database partition, Database partition, Containers, Tablespaces, Tablespaces, Tablespaces, Large objects (LOBs), Large objects (LOBs), Transaction logs, Transaction logs, Managing archive logs, Managing archive logs, Managing archive logs, The backup Command, Backup levels, Backup levels, Backup levels, Backup path and filenaming convention, Backup path and filenaming convention, Discovering the history of your backup operations, Automatic maintenance, Automatic maintenance, Crash recovery, Version recovery, Rollforward recovery, The restore Command, The rollforward Command, The recover Command, Performing an In-Place Version Recovery, Performing a Redirected Version Recovery, Performing a Rollforward Recovery, Performing a Rollforward Recovery, Reorganizing Data and Collecting Statistics
- architecture, DB2 Architecture, Managing archive logs, Instance, Databases, Schemas, Tables, Views, Indexes, DB2 engine dispatch units, System catalog tables, Database partition, Database partition, Containers, Tablespaces, Tablespaces, Tablespaces, Large objects (LOBs), Large objects (LOBs), Transaction logs, Transaction logs, Managing archive logs, Managing archive logs
  - archive logging, Managing archive logs
  - circular logging, Managing archive logs
  - container, Containers
  - database, Databases
  - database managed spaces (DMS), Tablespaces
  - engine dispatch units (EDUs), DB2 engine dispatch units
  - index, Indexes
  - instance, Instance
  - large (or long) tablespace, Large objects (LOBs)

large objects (LOBs), Large objects (LOBs)
partition, Database partition
partition group, Database partition
schema, Schemas
system catalog tables, System catalog tables
system managed spaces (SMS), Tablespaces
tables, Tables
tablespaces, Tablespaces
transaction logs, Transaction logs
view, Views
write ahead logging, Transaction logs
automated backup utilities, Automatic maintenance
backup command, The backup Command, Backup levels, Backup levels, Backup levels, Backup levels, Backup path and filenaming convention, Discovering the history of your backup operations
delta backup, Backup levels
full backup, Backup levels
incremental backup, Backup levels
pathname and filename, Backup path and filenaming convention
recovery history file, Discovering the history of your backup operations
crash recovery, Crash recovery
health monitor, Automatic maintenance
in-place version recovery, Performing an In-Place Version Recovery
offline backup, Managing archive logs
partial backups, Backup path and filenaming convention
recover command, The recover Command
redirected version recovery, Performing a Redirected Version Recovery
restore command, The restore Command
rollforward command, The rollforward Command
rollforward recovery, Rollforward recovery, Performing a Rollforward Recovery, Performing a Rollforward

Recovery
point in time (PIT), Performing a Rollforward Recovery
statistics, gathering, Reorganizing Data and Collecting Statistics
version recovery, Version recovery
DB2 UDB, IBM DB2 Backup and Recovery, Reorganizing Data and Collecting Statistics
dd utility, Backups change atime, Backing Up and Restoring with the dd Utility, Basic dd Options, Using dd to Copy a File or Raw Device, Using dd to Convert Data, Using dd to Determine the Block Size of a Tape, Using dd to Figure out the Backup Format
atime, changing, Backups change atime
backup format, Using dd to Figure out the Backup Format
convert data with, Using dd to Convert Data
copy a file or raw device with, Using dd to Copy a File or Raw Device
determine block size of a tape with, Using dd to Determine the Block Size of a Tape
options, Basic dd Options
de-duplication, De-Duplication Backup Systems, De-duplication
backup systems, De-Duplication Backup Systems
density versus compression, Density Versus Compression
development procedures (backups), following proper, Following Proper Development Procedures
differential backups, Backup Levels
digital audio tape (DAT), DDS drive
Digital Data Storage (DDS), DDS drive
Digital Linear Tape (DLT), DLT drives (end-of-lifed)
directories, cpio’s Restore Options, Torture-Testing Backup Programs, Pass I, Pass III
cpio restore, making needed, cpio’s Restore Options
data, writing (dump utility), Pass III

dump utility, evaluating for backup, Pass I
files, becoming (volatile filesystem backups), Torture-Testing Backup Programs
disasters, types of, Be Ready for Anything: 10 Types of Disasters, Be Ready for Anything: 10 Types of Disasters
disk drives, Be Ready for Anything: 10 Types of Disasters, Reliability
failure, losing data from, Be Ready for Anything: 10 Types of Disasters
versus tape media, Reliability
disk mirroring, System Recovery and Disk Mirroring
disk staging, Disk Targets
disk-as-disk targets, Disk-As-Disk Targets
disk-as-tape units (virtual tape libraries), Disk-As-Tape: Virtual Tape Libraries
disk-based backup, You Want to Learn About Disk-Based Backup
disk-to-disk-to-tape (D2D2T) backup, You Want to Learn About Disk-Based Backup
disks left powered off, Re-presentation
ditto utility, Backing Up and Restoring with the ditto Utility, Syntax of ditto When Backing Up, The Options to the ditto Command, Syntax of ditto when Restoring
options, The Options to the ditto Command
restoring with, Syntax of ditto when Restoring
syntax, Syntax of ditto When Backing Up
documentation, Take an Inventory, 12,000 gold pieces, Self-preservation: Document, document, document
backups, 12,000 gold pieces
importance in backup plan, Take an Inventory
restore program, importance of, Self-preservation: Document, document, document
downtime, cost of, What Will Downtime Cost You?
drive care, Drive Care
drives, backups, Bad or Dirty Drive or Tape
dump cycle, Backup Scheduling

dump utility, An Overview, Backing Up with the dump Utility, Using the index to create a table of contents, Backing Up with the dump Utility, Syntax of the dump Command, The Options to the dump Command, Interesting options for Solaris’s ufsdump utility, Specifying density and size (d and s), Different Versions of dump, Limitations of dump and restore, The dump utility, Torture-Testing Backup Programs, Demystifying dump, Demystifying dump, Dumpster Diving, Pass I, Pre-Pass III, Pass III, Pass IV, Summary of dump steps, A Final Analysis of dump, Determine the Blocking Factor
    backup data, writing, Pre-Pass III
    blocking factor, Determine the Blocking Factor
    demystifying, Demystifying dump
    differing versions, Different Versions of dump
    directories, writing data blocks, Pass III
    file data, dumping, Pass IV
    filesystems and, Demystifying dump
    how not to use, An Overview
    inconsistencies in backups, A Final Analysis of dump
    inodes, scanning filesystem for, Pass I
    limitations, Limitations of dump and restore
    Mac OS versus Linux, Backing Up with the dump Utility
    multiple volumes, avoiding, Specifying density and size (d and s)
    options, The Options to the dump Command, Interesting options for Solaris’s ufsdump utility
    steps, summary of, Summary of dump steps
    syntax, Syntax of the dump Command
    Unix platforms, differences, Dumpster Diving
    volatile filesystem backups, Torture-Testing Backup Programs
dump, ditto, tar, and cpio utilities, The tar, ditto, and cpio utilities
duty cycle, Duty Cycle
DVD recording formats, DVD recording formats

dye polymer recording method, Dye polymer recording method

E

ease of administration, Ease of Administration
electronic break-ins, losing data from, Be Ready for Anything: 10 Types of Disasters
electronic discovery requests, Satisfy electronic discovery requests
end-of-media (EOM) mark, Damaged Volume
endian-independent format, The Little Endian That Couldn’t
environments, protection levels, picking appropriate, Get the Coverage That You Need
Exchange, Database Structure, Exchange, Performing a basic restore, Exchange Architecture, Database Structure, Database Structure, Database Structure, Database Structure, Extensible Storage Engine, Extensible Storage Engine, Extensible Storage Engine, Stores, Stores, Stores, Storage Groups, Transaction Logfiles, Checkpoint Files, Reserve Logfiles, Circular Logging, Other Files, Single Instance Storage, Automatic Database Maintenance, Storage Limits, Backup, Verifying the Backup, Backup Strategy, Backup Types, Normal, Copy, Incremental, Differential, Daily, Exchange-specific, Windows-specific, Backup Methods, Online backups, Offline backups, Streaming backups, Shadow copy backups, Shadow copy backups, Shadow copy backups, Verifying backups, Using ntbackup to Back Up, Verifying the Backup, Restore, Performing a basic restore, Repair or Restore?, Exchange Repair, Overview, Restoring Exchange Mailbox or Public Folder Stores, Online database restore, Hard recovery versus soft recovery, Hard recovery versus soft recovery, Offline Database Restore, Recovery Storage Group,

Restoring deleted mailboxes, Restoring deleted items, Using ntbackup to Restore, Performing a basic restore
architecture, Exchange Architecture, Database Structure, Database Structure, Database Structure, Database Structure, Extensible Storage Engine, Extensible Storage Engine, Extensible Storage Engine, Stores, Stores, Stores, Storage Groups, Transaction Logfiles, Checkpoint Files, Reserve Logfiles, Circular Logging, Other Files, Single Instance Storage, Automatic Database Maintenance, Storage Limits
automatic database maintenance, Automatic Database Maintenance
checkpoint files, Checkpoint Files
circular logging, Circular Logging
Data Manipulation Language (DML), Extensible Storage Engine
database structure, Database Structure
ESE98, Database Structure
Extensible Storage Engine (ESE), Database Structure
Messaging Application Programming Interface (MAPI), Stores
Multipurpose Internet Mail Extensions (MIME), Stores
reserve logfiles, Reserve Logfiles
single instance storage, Single Instance Storage
storage groups, Storage Groups
storage limits, Storage Limits
stores, Database Structure, Stores
temporary files, Other Files
transaction logfiles, Transaction Logfiles
transactions, Extensible Storage Engine
version store, Extensible Storage Engine
backups, Backup, Verifying the Backup, Backup Strategy, Backup Types, Normal, Copy, Incremental, Differential, Daily, Exchange-specific, Windows-specific, Backup Methods, Online backups, Offline backups,

Streaming backups, Shadow copy backups, Shadow copy backups, Shadow copy backups, Verifying backups, Using ntbackup to Back Up, Verifying the Backup
clone, Shadow copy backups
copy, Copy
copy-on-write method, Shadow copy backups
daily, Daily
differential, Differential
Exchange-specific, Exchange-specific
incremental, Incremental
methods, Backup Methods
normal (full), Normal
ntbackup, Using ntbackup to Back Up, Verifying the Backup
offline (cold), Offline backups
online (hot), Online backups
strategy, Backup Strategy
streaming, Streaming backups
types, Backup Types
verifying, Verifying backups
Volume Shadow Copy Service, Shadow copy backups
Windows-specific, Windows-specific
repair, Exchange Repair
restore, Restore, Performing a basic restore, Repair or Restore?, Overview, Restoring Exchange Mailbox or Public Folder Stores, Online database restore, Hard recovery versus soft recovery, Hard recovery versus soft recovery, Offline Database Restore, Recovery Storage Group, Restoring deleted mailboxes, Restoring deleted items, Using ntbackup to Restore, Performing a basic restore
deleted items, Restoring deleted items
deleted mailboxes, Restoring deleted mailboxes
hard recovery, Hard recovery versus soft recovery
mailbox or public folder stores, Restoring Exchange Mailbox or Public Folder Stores

ntbackup, Using ntbackup to Restore
offline restore, Offline Database Restore
online database restore, Online database restore
overview, Overview
Recovery Storage Groups (RSGs), Recovery Storage Group
repair versus, Repair or Restore?
soft recovery, Hard recovery versus soft recovery
verifying, Performing a basic restore
versus other databases, Database Structure
exclude lists, Are You Backing Up What You Think You're Backing Up?
expansion, planning for, Plan for Expansion

F

files, Don't Forget Unix mtime, atime, and ctime, Preparing to Use mksysb and savevg, Torture-Testing Backup Programs, Torture-Testing Backup Programs, Pass I
backups, Preparing to Use mksysb and savevg
deleting, creating, shrinking (volatile filesystem backups), Torture-Testing Backup Programs
directories, becoming (volatile filesystem backups), Torture-Testing Backup Programs
dump utility, evaluating for backup, Pass I
time, Unix system recording for, Don't Forget Unix mtime, atime, and ctime
filesystems, Don't Go Overboard, Are You Backing Up What You Think You're Backing Up?, Test Everything!, Test Everything!, Volatile Filesystems, Question 1
backups, deciding what to include, Are You Backing Up What You Think You're Backing Up?
backups, testing, Test Everything!
changing during backup (dump), Question 1
snapshot backups, Don't Go Overboard
volatile, Volatile Filesystems (see volatile filesystems)

financial resources for backups, getting, Get the Money Your Backups Need
fire, Plan for the Worst, On-Site Storage
losses caused by, Plan for the Worst
media storage cabinets (fireproof), On-Site Storage
flash archive, Solaris Bare-Metal Recovery, Final Thoughts, Solaris Bare-Metal Recovery, Final Thoughts, Backup and Recovery Overview, Frequency of backup, Back up to disk or tape?, Back up to disk or tape?, Restore from tape or disk?, Interactive or noninteractive restore?, Other environmental constraints, Preparing for an Interactive Restore, Creating Flash Archive Images, Setup of a Noninteractive Restore, Creating a Noninteractive Tape Image, Creating a Noninteractive Disk Image, Post-Recovery Procedures
backup and recovery overview, Backup and Recovery Overview
bare-metal recovery using Solaris, Solaris Bare-Metal Recovery, Final Thoughts
creating, Creating Flash Archive Images
disk or tape, Back up to disk or tape?
environmental constraints, Other environmental constraints
flash image server, Back up to disk or tape?
frequency, Frequency of backup
interactive restore, Preparing for an Interactive Restore
interactive versus noninteractive restore, Interactive or noninteractive restore?
noninteractive disk image, Creating a Noninteractive Disk Image
noninteractive restore, Setup of a Noninteractive Restore
noninteractive tape image, Creating a Noninteractive Tape Image
post-recovery procedures, Post-Recovery Procedures
restore from tape or disk, Restore from tape or disk?

flash recovery area, Flash recovery area
flashback, Flashback
flexible restores, mksysb utility not supporting, IBM’s mksysb and savevg Utilities
format (backup volumes), finding, How Do I Read This Volume?
full backups, Backup Levels, Weekly schedule: All full/level 0 backups
    daily, Weekly schedule: All full/level 0 backups

G

G4L, Automate Bare-Metal Recovery with G4L, Setting Up G4L, Customizing G4L
    bare-metal recovery automated, Automate Bare-Metal Recovery with G4L
    customizing, Customizing G4L
    setting up, Setting Up G4L
Ghost 4 Linux, Automate Bare-Metal Recovery with G4L (see G4L)
Gigabit Ethernet, Gigabit Ethernet
GNU cpio, The Options to the cpio Command
GNU dump, format inconsistencies (native dump), handling, Different Backup Format
GNU tar, The Options to the tar Command, Conclusions, Different Backup Format
    format inconsistencies (native tar), handling, Different Backup Format
    volatile file system backups, Conclusions

H

hackers, tracking with ctime, Don’t Forget Unix mtime, atime, and ctime
hard disks, recovery companies, Disk Recovery Companies
hard links, Understanding Hard Links, Hard-Link Copies

copies, Hard-Link Copies
hardware, Be Ready for Anything: 10 Types of Disasters, System Cloning, How Do I Read This Volume?
compatibility problems, reading backup volumes, How Do I Read This Volume?
failure, losing data from, Be Ready for Anything: 10 Types of Disasters
recovering system onto new, System Cloning
headers, dumps, Pre-Pass III
Hierarchical Storage Management (HSM), Hierarchical Storage Management
holding disk, Holding Disk
hosts, complete list, maintaining, Are You Backing Up What You Think You're Backing Up?
hot backups, Physical and Logical Backups
hot site, Get the Coverage That You Need
HP Integrity versus HP9000 clients, Differences Between HP Integrity and HP9000 Clients
HP-UX bare-metal recovery, HP-UX Bare-Metal Recovery, System Recovery and Disk Mirroring

I

I/O operations, Block Size (Tape Volumes Only)
IBM, 3480 (end-of-lifed), 3590, 3592, TS1120, 3570 drive (a.k.a. Magstar MP), AIX Bare-Metal Recovery, AIX Bare-Metal Recovery, IBM's mksysb and savevg Utilities
3480, 3480 (end-of-lifed)
3570, 3570 drive (a.k.a. Magstar MP)
3590, 3590
3592, 3592
AIX operating system, AIX Bare-Metal Recovery
bare-metal recovery tools, AIX Bare-Metal Recovery
mksysb utility, IBM's mksysb and savevg Utilities
T1120, TS1120

IBM DB2 Universal Database, IBM DB2 Backup and Recovery (see DB2)
Ignite-UX recovery tool, HP-UX Bare-Metal Recovery, Troubleshooting Recovery Operations, Ignite-UX Overview, Network Services and Remote Boot Protocols, Differences Between HP Integrity and HP9000 Clients, Planning for Ignite-UX Archive Storage and Recovery, Recovery Archive Management, Considerations for the Remote Booting of Clients, Sizing the Recovery Archive, Configuring an Ignite-UX Network Server, Recovery Archive Management, Verifying Archive Contents, Troubleshooting Recovery Operations, Security, System Recovery and Disk Mirroring
  archive contents, verifying, Verifying Archive Contents
  archive management, Recovery Archive Management
  disk mirroring, System Recovery and Disk Mirroring
  HP Integrity versus HP9000 clients, Differences Between HP Integrity and HP9000 Clients
  network server, configuring, Configuring an Ignite-UX Network Server
  network services and remote boot protocols, Network Services and Remote Boot Protocols
  overview, Ignite-UX Overview
  planning for archive storage and recovery, Planning for Ignite-UX Archive Storage and Recovery, Recovery Archive Management
  recovery archive, sizing, Sizing the Recovery Archive
  remote booting, Considerations for the Remote Booting of Clients
  security issues, Security
  troubleshooting, Troubleshooting Recovery Operations
image versus filesystem level, Image level or filesystem level?
image.data file (AIX system backup), Preparing for the restore
include lists, backups, Are You Backing Up What You Think You’re Backing Up?

incremental backups, Backup Levels, IBM’s mksysb and savevg Utilities
    AIX mksysb utility, not supporting, IBM’s mksysb and savevg Utilities
index, Index
Information Lifecycle Management (ILM), Information Lifecycle Management
Informix, Informix
initialization parameters, Initialization parameters
inode number, Understanding Hard Links
inodes, dump utility handling, Pass I
input mode, informing cpio about, cpio’s Restore Options
installation, systems (new), backups as integral part, Strategy: Make backups an integral part of the installation process
instance, Instance, Instance
    defined, Instance
    versus server, Instance
integrated VTL, How do you eject virtual tapes?
Intel systems, bare-metal recovery of, Linux and Windows, Commercial Solutions
Intel-based Mac, Preparing for a Bare-Metal Recovery
interactive restore, flash archive, Preparing for an Interactive Restore
interleaving, Simultaneous Backup of Many Clients to One Drive
inventory, backup volumes, Take an Inventory, 12,000 gold pieces, Choosing a media vaulting vendor
    checks and spot checks, 12,000 gold pieces
    storage vendor, cross-checking locations, Choosing a media vaulting vendor
iPod, using for backup, Preparing for a Bare-Metal Recovery

J

journaling filesystems, Be Ready for Anything: 10 Types of Disasters
jukebox, Automated Backup Hardware
justifying financial needs (backups), Be ready

K

Knoppix, using to perform bare-metal recovery of Intel systems, Four backup options

L

LAN-free backup, LAN-Free Backup
large object (LOB) data, LOBs, Large object datatypes
Laser Magnetic Storage (LMS) NCTP drive, LMS NCTP drive
levels (backups), Backup Levels, “In the Middle of the Night…”, Which Levels Do You Run and When?
    scheduling, Which Levels Do You Run and When?
linear serpentine recording method, Helical and Linear Tape Drives Are Different
Linear Tape Open (LTO), LTO drives
Linux bare-metal recovery, Linux and Windows, Commercial Solutions, Live or alternate boot?, Image level or filesystem level?, Complete disk or separate partitions?, Alt-Boot Full Image Method, Alt-Boot Partition Image Method, Alt-Boot Filesystem Method, Automate Bare-Metal Recovery with G4L
    alt-boot filesystem method, Alt-Boot Filesystem Method
    alt-boot full image method, Alt-Boot Full Image Method
    alt-boot method, Live or alternate boot?, Image level or filesystem level?, Complete disk or separate partitions?
        complete disk versus separate partitions, Complete disk or separate partitions?
        image versus filesystem level, Image level or filesystem level?

alt-boot partition image method, Alt-Boot Partition Image Method
with G4L, Automate Bare-Metal Recovery with G4L
little-endian platforms, The Little Endian That Couldn’t
live method, Live Method
live method for backups and restores, Live Method
live versus alternative boot, Live or alternate boot?, Live or alternate boot?
LiveCD Linux distribution, Live or alternate boot?
logical elements of databases, The Power User’s View: Logical Elements of a Database, Checkpoint
logical logs, Transaction log
logical volumes, map file, creating for each, Preparing for the restore
losses (data), You Never Want to Say These Words, What Will Lost Data Cost You?
cost, determining, What Will Lost Data Cost You?
unacceptability of, You Never Want to Say These Words
LVM, backing up configuration information, Take an Inventory

M

Mac OS X backups, How Mac OS Filesystems Are Different
Mac OS X bare-metal recovery, Mac OS X Bare-Metal Recovery, Recover the System
Mac, Intel-based, Preparing for a Bare-Metal Recovery
Macintosh Hierarchical File System, How Mac OS Filesystems Are Different
magneto-optical (MO), Why the Word “Volume” Instead of “Tape”?, Magneto-optical recording method, Magneto-optical recording format
media, disks, as backup medium, Why the Word “Volume” Instead of “Tape”?
recording format, Magneto-optical recording format
recording method, Magneto-optical recording method

mail storage formats, A brief word about mail formats
Mammoth drives, Mammoth drive (end-of-lifed)
managing the intelligence behind the backup system, Cost
map file, creating for each logical volume, Preparing for the restore
master boot record (MBR), Complete disk or separate partitions?
master database, Master database
mean-time-between-failure (MTBF), Reliability
media, Take an Inventory, Storage in General, How It Works, Wrong Media Type
    backup volumes, Storage in General
    booting from alternative, How It Works
    inventorying, Take an Inventory
    types (different), problems reading backups, Wrong Media Type
media life, passes, How Often Should I Change My Media?
Media Recognition System (MRS), Wrong Media Type
metadata, backing up, Step 1: Back Up Important Metadata
Microsoft Exchange, Exchange Architecture (see Exchange)
Microsoft Windows, Exchange Architecture (see Windows)
midrange tape drive types, Midrange Tape Drive Types
mirroring, Get the Coverage That You Need, Be Ready for Anything: 10 Types of Disasters
    disk drive failure, protecting against, Be Ready for Anything: 10 Types of Disasters
    multiple-site of database servers, Get the Coverage That You Need
mksysb utility, IBM’s mksysb and savevg Utilities, IBM’s mksysb and savevg Utilities, IBM’s mksysb and savevg Utilities, Setting a mksysb Definition for a Client
mksysb utility AIX, AIX Bare-Metal Recovery, IBM’s mksysb and savevg Utilities, Backing Up rootvg to Locally Attached Tape, Verifying a mksysb or savevg Backup

image, checking and restoring data from, Verifying a mksysb or savevg Backup
limitations of, IBM’s mksysb and savevg Utilities
using, Backing Up rootvg to Locally Attached Tape
mkszfile utility, Preparing for the restore, Backing Up rootvg to a Remote Tape Drive
client, running on, Backing Up rootvg to a Remote Tape Drive
generating image.data file, Preparing for the restore
MLR drives, MLR 1-3 drives
modification time, Don’t Forget Unix mtime, atime, and ctime
money for backups, getting, Get the Money Your Backups Need
monitoring backups, Monitoring Your Backups
morale, backup and, What Will Lost Data Cost You?
mounted filesystems (idle), dumping, Question 4
movement-tracking, backup volumes, 12,000 gold pieces
MRS (Media Recognition System), Wrong Media Type
mtime, Don’t Forget Unix mtime, atime, and ctime, cpio’s Restore Options
cpio restores, restoring original, cpio’s Restore Options
multiplexing (mirroring), Disk drives: Very flexible, Control file
multistreaming, Simultaneous Backup of One Client to Many Drives
MySQL, MySQL, Restoring a MySQL cluster, MySQL, MySQL, MySQL Architecture, Instances, Startup scripts, Databases and tablespaces, Databases and tablespaces, Large objects, Binary log, MyISAM Storage Engine, InnoDB Storage Engine, Transactions, Tablespace, Datafile, Rollback segment or log group, Rollback segment or log group, Transaction log, Other Storage Engines, Other Storage Engines, Other Storage Engines, Other Storage Engines, MySQL Backup and Recovery Methodologies, SQL-Level Backup and Recovery, Backing up MyISAM

tables, Backing up InnoDB tables, File-Level Backup and Recovery, File-level backup with mysqlhotcopy, Using Point-in-Time Recovery
architecture, MySQL Architecture, Instances, Startup scripts, Databases and tablespaces, Databases and tablespaces, Large objects, Binary log
binary log, Binary log
databases, Databases and tablespaces
instance, Instances
large objects (LOBs), Large objects
startup scripts, Startup scripts
tablespaces, Databases and tablespaces
backup and recovery methodologies, MySQL Backup and Recovery Methodologies, SQL-Level Backup and Recovery, Backing up MyISAM tables, Backing up InnoDB tables, File-Level Backup and Recovery, File-level backup with mysqlhotcopy, Using Point-in-Time Recovery
file-level, File-Level Backup and Recovery
InnoDB tables, Backing up InnoDB tables
MyISAM tables, Backing up MyISAM tables
mysqlhotcopy, File-level backup with mysqlhotcopy
point-in-time, Using Point-in-Time Recovery
SQL-level, SQL-Level Backup and Recovery
pluggable storage engine architecture, MySQL
storage engines, MySQL, MyISAM Storage Engine, InnoDB Storage Engine, Transactions, Tablespace, Datafile, Rollback segment or log group, Rollback segment or log group, Transaction log, Other Storage Engines, Other Storage Engines, Other Storage Engines, Other Storage Engines
datafiles, Datafile
Falcon, Other Storage Engines
InnoDB, InnoDB Storage Engine
log group, Rollback segment or log group
MyISAM, MyISAM Storage Engine

NDB, Other Storage Engines
PBXT, Other Storage Engines
rollback segment, Rollback segment or log group
SolidDB, Other Storage Engines
tablespaces, Tablespace
transaction logs, Transaction log
transactions, Transactions

N

named pipes, cat contents to tapeserver tape drive, Backing Up rootvg to a Remote Tape Drive
NAS disk-as-disk targets, NAS disk-as-disk targets
native backup utilities, Torture-Testing Backup Programs, Different Backup Format
  incompatible versions, Different Backup Format
  testing, Torture-Testing Backup Programs
native transfer speed, Transfer Speed
near-CDP, Open-Source Near-CDP, Windows, Mac OS X, and the Future, Near-Continuous Data Protection Systems
near-continuous data protection, Open-Source Near-CDP (see near-CDP)
nearline storage, Nearline and Offline Storage
network traffic, Reduction in Network Traffic
NFS filesystems, scripts for installation, Take an Inventory
NIM (Network Install Manager), Setting Up NIM
NIS+, Basic Sizing
NIS, scripts for installation, Take an Inventory
noninteractive disk image, flash archive, Creating a Noninteractive Disk Image
noninteractive restore, flash archive, Setup of a Noninteractive Restore
noninteractive tape image, flash archive, Creating a Noninteractive Tape Image
notification, Notification

ntbackup, Backing Up and Restoring with ntbackup, Creating a Simple Backup Configuration, Executing Your Simple Backup, Restoring with ntbackup, Using ntbackup to Back Up
creating a simple backup, Creating a Simple Backup Configuration
Exchange and, Using ntbackup to Back Up
executing, Executing Your Simple Backup
restoring from, Restoring with ntbackup
number of passes, and media life, How Often Should I Change My Media?

O

off-site storage, Be Ready for Anything: 10 Types of Disasters, Off-Site Storage
methods, Off-Site Storage
natural disasters, preparedness for, Be Ready for Anything: 10 Types of Disasters
offline storage, Nearline and Offline Storage
on-site storage, On-Site Storage
ones complement platforms, The Little Endian That Couldn’t
open-source backup, You’re Curious About Open-Source Backup Products
operating systems, Don’t Go Overboard, Strategy: Make backups an integral part of the installation process, AIX Bare-Metal Recovery, System Cloning, AIX and Its 512-Byte Block Size
AIX system, AIX Bare-Metal Recovery
backup tools, including, Don’t Go Overboard
block size, hardcoding, AIX and Its 512-Byte Block Size
cloning to different system, System Cloning
new versions, affect on backups, Strategy: Make backups an integral part of the installation process

optical drives, Optical drives: A little more flexible, Optical Drives, UDO recording format, Optical Drives, Optical Drives
    recording format, Optical Drives
    recording method, Optical Drives
Oracle architecture, Oracle Architecture, Finding All Instances, Instance, Database, Table, Index, Large object datatypes, Object, Row, Attribute, Blocks, Extents, Segment, Datafile, Datafile, Tablespace, Partition, Control file, Control file, Transaction, Undo tablespace, Checkpoint, Flash recovery area, Redo log, Redo log, Initialization parameters, Initialization parameters, Restore versus recover
    attribute, Attribute
    block, Blocks
    checkpoint, Checkpoint
    control file, Control file
    database, Database
    datafile, Datafile
    extent, Extents
    flash recovery area, Flash recovery area
    index, Index
    initialization parameters, Initialization parameters
    instance, Instance
    large object (LOB) data, Large object datatypes
    log group, Redo log
    multiplexing (mirroring), Control file
    object, Object
    partitions, Partition
    redo log, Redo log
    restore versus recovery, Restore versus recover
    row, Row
    segment, Segment
    server parameter file (spfile), Initialization parameters
    system change number (SCN), Datafile
    table, Table

tablespace, Tablespace
transaction, Transaction
undo tablespace, Undo tablespace
Oracle backup and recovery, Don’t Go Overboard, Are You Backing Up What You Think You’re Backing Up?, Oracle Backup and Recovery, A Broken Record, Finding All Instances, Physical Backups Without rman, Debunking Hot-Backup Myths, Cold Backup, Hot Backup, Debunking Hot-Backup Myths, Physical Backups with rman, The database.inc.rman command file (level 1 backups), Flashback, Other Commercial Backup Methods, Managing the Archived Redo Logs, Recovering Oracle, You’re Done!, Logical Backups
archived redo logs, managing, Managing the Archived Redo Logs
cold backup, Cold Backup
flashback, Flashback
hot backup, Hot Backup, Debunking Hot-Backup Myths
instances, finding all, Finding All Instances
logical backups, Logical Backups
Oracle databases, Don’t Go Overboard, Are You Backing Up What You Think You’re Backing Up?
backing up, Don’t Go Overboard
instances, listing for backups, Are You Backing Up What You Think You’re Backing Up?
recovery guide, Recovering Oracle, You’re Done!
rman, Physical Backups with rman, The database.inc.rman command file (level 1 backups)
SQL Backtrack, Other Commercial Backup Methods
user-managed backups, Physical Backups Without rman, Debunking Hot-Backup Myths
ownership, changing with ctime, Don’t Forget Unix mtime, atime, and ctime

P

page, logical versus physical log, Transaction log
paper copies, media inventory, importance of, Take an Inventory
parallelism, Simultaneous Backup of Many Clients to One Drive
partition, Partition
partitioning your OS drive, Take an Inventory, Choosing Backup Methods
    recording information on, Take an Inventory
partitions, Partition, Multiple Partitions on a Tape
    backup tape with multiple, Multiple Partitions on a Tape
pax (portable archive exchange), pax
PEBuilder web site, Bare-Metal Recovery
peripherals, inventorying devices, Take an Inventory
permissions, changing with ctime, Don’t Forget Unix mtime, atime, and ctime
personnel, backups and, The Impossible Job That No One Wants
phase change recording method, Phase change recording method
Phillips LMS NCTP, LMS NCTP drive
philosophy of backup, the, The Philosophy of Backup, Why the Word “Volume” Instead of “Tape”?
physical log, Transaction log
planning for worst-case scenarios, Plan for the Worst
platforms, Prepare in Advance, The Little Endian That Couldn’t
    backup volumes, reading from different, Prepare in Advance
    little- or big-endian, The Little Endian That Couldn’t
PostgreSQL, PostgreSQL, Restoring from a Point-in-Time Backup, PostgreSQL Architecture, Write Ahead Log, Clusters, Tablespace, Tablespace, Startup Scripts, System Tables, Large Objects, Rollback Process, Write Ahead Log, Backup and Recovery, Restoring from a Point-in-Time Backup, Backing up with pg_dump, Restoring with

pg_restore, Using pg_dump with psql, Using pg_dumpall with psql, Point-in-Time Recovery
architecture, PostgreSQL Architecture, Write Ahead Log, Clusters, Tablespace, Tablespace, Startup Scripts, System Tables, Large Objects, Rollback Process, Write Ahead Log
clusters, Clusters
large objects (LOBs), Large Objects
pagefiles (datafiles), Tablespace
rollback process, Rollback Process
startup scripts, Startup Scripts
system tables, System Tables
tablespace, Tablespace
write ahead log (WAL), Write Ahead Log
backup and recovery, Backup and Recovery, Restoring from a Point-in-Time Backup, Backing up with pg_dump, Restoring with pg_restore, Using pg_dump with psql, Using pg_dumpall with psql, Point-in-Time Recovery
pg_dump, Backing up with pg_dump
pg_dump with psql, Using pg_dump with psql
pg_dumpall with psql, Using pg_dumpall with psql
pg_restore, Restoring with pg_restore
point-in-time (PIT), Point-in-Time Recovery
profile file, flash archive, profile
psync, rsyncx, hfstar, xtar, and hfspax, psync, rsyncx, hfstar, xtar, and hfspax

R

RAID, levels of, Be Ready for Anything: 10 Types of Disasters
raw partitions, dump utility, accessing files through, Demystifying dump
RDBMS, What Can Happen to an RDBMS? (see databases)
rdiff-backup utility, rdiff-backup, Windows, Mac OS X, and the Future, rdiff-backup, rdiff-backup, Quick Start,

Windows, Mac OS X, and the Future
diffs or deltas, rdiff-backup
metadata and, rdiff-backup
platforms, Windows, Mac OS X, and the Future
quick-start example, Quick Start
re-presentation, Content-awareness
read-tape.sh script, reading bad tape with, Reading a “Flaky” Tape
reading backup volumes, How Do I Read This Volume?
recording format, Optical Drives, Optical Recording Formats
recording method, Optical Drives
records, gaps between (tape backups), Block Size (Tape Volumes Only), Block Size (Tape Volumes Only)
recovery, Don’t Go Overboard, Ease of Recovery
ease, Ease of Recovery
requirements, Don’t Go Overboard
redo log, Redo log
Redundant Array of Inexpensive Tapes (RAIT), Device Management
referential integrity problems, Referential Integrity Problems
regulatory bodies, backups and, 12,000 gold pieces
remote devices, mksysb backup, performing to, Backing Up rootvg to a Remote Tape Drive
remote office backup, Aggressive Requirements, Remote Office Backup
remote restores, mksysb utility, not supporting, IBM’s mksysb and savevg Utilities
removability, Removability
replication, Replication, Replication
repositioning, Simultaneous Backup of Many Clients to One Drive, Tape Drives Must Be Streamed
tape drives, Tape Drives Must Be Streamed
resource fork, Don’t Forget Mac OS Resource Forks

restore utility, Restoring with the restore Utility, Specifying no query during restore (y), Blocking Factor, Syntax of the restore Command, The Options to the restore Command, Specifying no query during restore (y), Limitations of dump and restore
    blocking factor workaround, Blocking Factor
    limitations, Limitations of dump and restore
    options, The Options to the restore Command, Specifying no query during restore (y)
    syntax, Syntax of the restore Command
restores, disk recovery companies, Disk Recovery Companies
restoring to larger hard drives, Step 4: Restore the Boot Block Information
rman, Physical Backups with rman, The database.inc.rman command file (level 1 backups), Important New rman Features, Automating rman
    automating, Automating rman
    features, new, Important New rman Features
rollback logs, Rollback log, Transaction log
    transaction log versus, Transaction log
root volume group, Preparing to Use mksysb and savevg, Backing Up rootvg to Locally Attached Tape
    filesystems, backing up with mksysb, Backing Up rootvg to Locally Attached Tape
    preparing for mksysb utility builds of multiple systems, Preparing to Use mksysb and savevg
row, Row
RPO (recovery point objective), Aggressive Requirements
rsh utility, mksysb, performing to remote device, Backing Up rootvg to a Remote Tape Drive
rsnapshot utility, rsnapshot, Platform Support, When Not to Use rsnapshot, Setting Up rsnapshot, The rsnapshot Community
    community, The rsnapshot Community
    platform support, Platform Support

setting up, Setting Up rsnapshot
when not to use, When Not to Use rsnapshot
rsync utility, Using rsync, Basic rsync Syntax, Restoring with rsync, rsync with Snapshots, Atomicity and partial transfers, Beyond the Example
restoring with, Restoring with rsync
syntax, Basic rsync Syntax
with snapshots, rsync with Snapshots, Atomicity and partial transfers
RTO (recovery time objective), Aggressive Requirements
rules file, flash archive, The rules file

S

SAN disk-as-disk targets, SAN disk-as-disk targets
savevg utility, IBM’s mksysb and savevg Utilities, mksysb and savevg Format, savevg Operations
schadenfreude, Schadenfreude
scripts, expansion, planning for, Plan for Expansion
security, Be Ready for Anything: 10 Types of Disasters, Security
breaches of, Be Ready for Anything: 10 Types of Disasters
segment, Segment
segments, Tablespace
server parameter file (spfile), Initialization parameters
server-free (serverless) backup, Server-Free (or Serverless) Backup
servers, automating installation of new, Take an Inventory
shoe-shining, of tape drives, How Often Should I Change My Media?
silo versus library, Automated Backup Hardware
simplicity in backups, preserving, Keep It Simple, SA
smit menus, mksysb, using through, Backing Up rootvg to Locally Attached Tape

snapshots, Don’t Go Overboard, Using Snapshots to Back Up a Volatile Filesystem
    using as backups, Don’t Go Overboard
    volatile filesystem backups, Using Snapshots to Back Up a Volatile Filesystem
software failure, losing data from, Be Ready for Anything: 10 Types of Disasters
Solaris bare-metal recovery, Solaris Bare-Metal Recovery, Final Thoughts, Solaris Bare-Metal Recovery, Final Thoughts
    flash archive, Solaris Bare-Metal Recovery, Final Thoughts
Sony AIT, AIT drive
Sony DTF, DTF drive
speed of recovery, options affecting, What Will Downtime Cost You?
SQL Backtrack, Other Commercial Backup Methods
SQL Server, SQL Server, Master Database Restore, Overview of SQL Server, Connecting to and Administering SQL Server, SQL Server Authentication, Instance, Databases, Databases, System databases, User databases, Tables, System tables, System tables, Temporary tables, Index, Partitioned tables, Partitioned indexes, Stored Procedures, Memory Management, Database Files, Filegroups, Transaction Log, Pages, Extents, Partitions, Partitions, Partitions, Table and Index Specifics, Table and Index Specifics, Snapshot Backups (2005), Backups, Logical (Table-Level) Backups, Backup Devices, Logical and physical devices, Logical and physical devices, Recovery Models, Full, Differential, Transaction log, Copy-only backup, Partial backups, Partial backups, File and filegroup, Backup/Restore of System Databases, Verify Backups, Backup Expiration Date, How to Back Up, Command-line backup with Transact-SQL, Transaction Log Backups, Log truncation, Master Database Backups, Scheduling a Backup, Logical (Table-Level) Backups,

Restore and Recovery, Master Database Restore, Restore and Recovery, Restore and Recovery, Components of a Restore, Components of a Restore, Components of a Restore, Components of a Restore, Components of a Restore, Recovery Roadmap, Database Restore, Command-line restore with Transact-SQL, Master Database Restore
administering, Connecting to and Administering SQL Server
architecture, Overview of SQL Server, Instance, Databases, Databases, Index, Partitioned tables, Partitioned indexes, Memory Management, Database Files, Filegroups, Pages, Extents, Partitions, Partitions, Table and Index Specifics
database, defined, Databases
datafiles, Database Files
extents, Extents
filegroup, Filegroups
heap data, Partitions
indexes, Index, Table and Index Specifics
instance, Instance
master databases, Databases
memory management, Memory Management
overview, Overview of SQL Server
pages, Pages
partitioned indexes, Partitioned indexes
partitioned tables, Partitioned tables
partitions, Partitions
authentication, SQL Server Authentication
backups, Backups, Logical (Table-Level) Backups, Backup Devices, Logical and physical devices, Logical and physical devices, Recovery Models, Full, Differential, Transaction log, Copy-only backup, Partial backups, Partial backups, File and filegroup, Backup/Restore of System Databases, Verify Backups, Backup Expiration Date, How to Back Up, Command-line backup with Transact-SQL, Transaction Log Backups,

Log truncation, Master Database Backups, Scheduling a Backup, Logical (Table-Level) Backups
backup devices, Backup Devices
copy-only, Copy-only backup
differential, Differential
differential partial, Partial backups
expiration date, Backup Expiration Date
files or filegroups, File and filegroup
full (procedure), How to Back Up
full database, Full
log truncation, Log truncation
logical (table-level), Logical (Table-Level) Backups
logical name, Logical and physical devices
master database, Master Database Backups
partial, Partial backups
physical name, Logical and physical devices
recovery models, Recovery Models
scheduled, Scheduling a Backup
system databases, Backup/Restore of System Databases
Transact-SQL, command-line backup, Command-line backup with Transact-SQL
transaction logs, Transaction log, Transaction Log Backups
verifying, Verify Backups
cluster data, Partitions
restore and recovery, Restore and Recovery, Master Database Restore, Restore and Recovery, Restore and Recovery, Components of a Restore, Components of a Restore, Components of a Restore, Components of a Restore, Components of a Restore, Recovery Roadmap, Database Restore, Command-line restore with Transact-SQL, Master Database Restore
backup recovery, Components of a Restore
components of, Components of a Restore
data copy, Components of a Restore

full backup restore, Database Restore
master database restore, Master Database Restore
recovery, Restore and Recovery
restore, Restore and Recovery
restore sequence, Components of a Restore
roadmap, Recovery Roadmap
roll forward set, Components of a Restore
Transact-SQL, command-line restore, Command-line restore with Transact-SQL
snapshot backups, Snapshot Backups (2005)
stored procedures, Stored Procedures
system databases, System databases
tables, Tables, System tables, System tables, Temporary tables, Table and Index Specifics
dedicated administrator connect (DAC), System tables
system, System tables
temporary, Temporary tables
transaction logs, Transaction Log
user databases, User databases
ssh or rsh as a conduit between systems, Using ssh or rsh as a Conduit Between Systems
stacker, Automated Backup Hardware
stacking, Stacking
storage area network (SAN), LAN-Free Backup, Storage Area Networks
storage containers for media, On-Site Storage
storage management, Storage Management Features, Information Lifecycle Management
storage vendors, testing quality of procedures, Testing your chosen vendor
storing backups, Storing Your Backups
streaming tape drives, Simultaneous Backup of Many Clients to One Drive
Super DLT, DLT-S drives (aka Super DLT)

swapping bytes, incompatibility problems, The Little Endian That Couldn’t
Sybase, Sybase Backup and Recovery, Sybase Backup and Recovery, Bring the database online, Sybase Backup and Recovery, Sybase Backup and Recovery, Sybase Architecture, Sybase Command-Line Utilities, isql, bcp, dsedit, dsedit, Required Environment Variables, Server, Server, Server, Engine, Database, Transaction, Table, System Table, System Table, Index, Stored Procedures, Page, Extent, Datafiles and Devices, Datafiles and Devices, Segment, Configuration File, Transaction Log, Transaction log sizing, The interfaces File, The SYBASE.sh and SYBASE.csh Files, Backup Server, Dump Device, Hot and Cold Backups, Protecting Your Database, Have a Run Book, dbcc: The Database Consistency Checker, Reorgs, Update Statistics, Configuration Audits, Implement Mirroring and Disk Striping, Implement Mirroring and Disk Striping, How to Back Up Your Servers, Backup striping and compression, Backup striping and compression, Backup striping and compression, Have a Run Book, Backup Automation Through Scripting, Auditing using bcp, Logical Backups, Logical Backups, Performing a logical restore, Performing a logical restore, Auditing using bcp, Physical Backups with a Storage Manager, Recovering from a Disaster, Restoring from Backups, Common Sybase Procedures, Reload your database, Procedure 1: How to Start Sybase, Procedure 3: How to Shut Down Your Server, Procedure 4: How to Set Server Configuration Options, Procedure 5: How to Set Database-Level Options, Procedure 6: How to Run a Query, Sybase Recovery Procedure, Check your available free space
architecture, Sybase Architecture, Server, Engine, Database, Transaction, Table, System Table, System Table, Index, Stored Procedures, Page, Datafiles and Devices, Segment, Transaction Log, Transaction log

sizing, The interfaces File, The SYBASE.sh and SYBASE.csh Files, Backup Server, Dump Device
backup server, Backup Server
Data Manipulation Language (DML), System Table
database, Database
database index, Index
datafiles (devices), Datafiles and Devices
dataserver process, Server
engine, Engine
environment files, The SYBASE.sh and SYBASE.csh Files
interfaces file, The interfaces File
logical dump devices, Dump Device
page, Page
segment, Segment
stored procedure, Stored Procedures
system table, System Table
table, Table
transaction log, Transaction Log
transaction log sizing, Transaction log sizing
transactions, Transaction
auditing, Auditing using bcp
backup and recovery, Sybase Backup and Recovery, Bring the database online
backup automation through scripting, Backup Automation Through Scripting, Auditing using bcp
backupserver process, Server
bcp utility, Performing a logical restore
command-line utilities, Sybase Command-Line Utilities, isql, bcp, dsedit
bcp command, bcp
dsedit utility, dsedit
isql command, isql
configuration file, Configuration File
disaster recovery, Recovering from a Disaster
dsync devices, Datafiles and Devices

dump command, Logical Backups
environment variables, Required Environment Variables
extent, Extent
hot and cold backups, Hot and Cold Backups
interfaces file, dsedit
logical backups, Logical Backups
logical restore, Performing a logical restore
maintenance scripts, Sybase Backup and Recovery
physical backups with storage manager, Physical Backups with a Storage Manager
protecting, Protecting Your Database, Have a Run Book, dbcc: The Database Consistency Checker, Reorgs, Update Statistics, Configuration Audits, Implement Mirroring and Disk Striping, Implement Mirroring and Disk Striping, How to Back Up Your Servers, Backup striping and compression, Backup striping and compression, Backup striping and compression, Have a Run Book
    backing up servers, How to Back Up Your Servers
    configuration audit, Configuration Audits
    dbcc (Database Consistency Checker), dbcc: The Database Consistency Checker
    disk striping, Implement Mirroring and Disk Striping
    dump command, Backup striping and compression
    file compression, Backup striping and compression
    mirroring, Implement Mirroring and Disk Striping
    reorganizing table data, Reorgs
    run book, Have a Run Book
    striping, Backup striping and compression
    update statistics, Update Statistics
recovery procedures, Common Sybase Procedures, Reload your database, Procedure 1: How to Start Sybase, Procedure 3: How to Shut Down Your Server, Procedure 4: How to Set Server Configuration Options, Procedure 5: How to Set Database-Level Options,

Procedure 6: How to Run a Query, Sybase Recovery Procedure, Check your available free space
checking available free space, Check your available free space
configuration options, setting, Procedure 4: How to Set Server Configuration Options
database-level options, setting, Procedure 5: How to Set Database-Level Options
diagnosing, Sybase Recovery Procedure
query, running, Procedure 6: How to Run a Query
shutting down server, Procedure 3: How to Shut Down Your Server
starting, Procedure 1: How to Start Sybase
restoring from backups, Restoring from Backups
server, Server
server documentation, Sybase Backup and Recovery
Sybase Adaptive Server Enterprise, Server (see Sybase)
Sysback utility, AIX Bare-Metal Recovery
sysidcfg file, flash archive, The sysidcfg file
system change number (SCN), Datafile
system cloning, System Cloning, System Cloning
System Independent Data Format (SIDF), What happened to SIDF?
system recovery with Ignite-UX tool, System Recovery with Ignite-UX
System Restore (Windows), Using System Restore in Windows, Creating Restore Points, Recovering Windows Using a Restore Point
creating restore points, Creating Restore Points
recovering using a restore point, Recovering Windows Using a Restore Point
systems, Don’t Skip This Chapter!, Take an Inventory, Back Up All or Part of the System?, Be Ready for Anything: 10 Types of Disasters, Strategy: Make backups an integral part of the installation process, Prepare in Advance

administrators, unfamiliar with backups, Don’t Skip This Chapter!
backing up whole or part, Back Up All or Part of the System?
familiarity with, importance in backup/recovery, Take an Inventory
new, adding to backup list, Strategy: Make backups an integral part of the installation process
reading backup volumes from different, Prepare in Advance
staff error, losing data from, Be Ready for Anything: 10 Types of Disasters
systemwide failure, losing data from, Be Ready for Anything: 10 Types of Disasters

T

table, Table
table of contents, listing for mksysb tape, Verifying a mksysb or savevg Backup
tablespace, Extents
tape devices, drives, different (problems with reading backups), Wrong Media Type
tape drives, Duty Cycle, How Often Should I Change My Media?, Tape Drives, VXA, Tape Drives Must Be Streamed, Tape Drives Must Be Streamed, Variable Speed Tape Drives, Helical and Linear Tape Drives Are Different, Helical and Linear Tape Drives Are Different, Cartridges Versus Cassettes, Midrange Tape Drive Types, TS1120, 9840 drives, 9940 drives, T10000 drives, Backing Up rootvg to a Remote Tape Drive
9840, 9840 drives
9940, 9940 drives
cartridge versus cassette, Cartridges Versus Cassettes
helical versus linear, Helical and Linear Tape Drives Are Different

linear serpentine recording method, Helical and Linear Tape Drives Are Different
midrange types, Midrange Tape Drive Types
mksysb backup, performing to, Backing Up rootvg to a Remote Tape Drive
repositioning, Tape Drives Must Be Streamed
shoe-shining, How Often Should I Change My Media?
streaming, Tape Drives Must Be Streamed
T10000, T10000 drives
T1120, TS1120
variable-speed, Variable Speed Tape Drives
tape host, Client/Server Architecture Using Nonproprietary Tools
tapes, backups, Why the Word “Volume” Instead of “Tape”?, mksysb and savevg Format
tar utility, Backups change atime, Backing Up and Restoring with the tar Utility, Restoring to an alternate location, The Syntax of tar When Backing Up, Torture-Testing Backup Programs, Torture-Testing Backup Programs, Demystifying dump, Determine the Blocking Factor
atime, changing, Backups change atime
blocking factor, determining, Determine the Blocking Factor
filesystem-based, Demystifying dump
syntax, The Syntax of tar When Backing Up
volatile filesystem backups, Torture-Testing Backup Programs, Torture-Testing Backup Programs
target throughput rate, streaming, Compression Makes It Harder to Stream Drives
testing backup utilities, Torture-Testing Backup Programs, Torture-Testing Backup Programs
volatile filesystems, Torture-Testing Backup Programs
testing backups, Testing Your Backups
testing development procedures (backups), Following Proper Development Procedures

theft, losing data from, Be Ready for Anything: 10 Types of Disasters
throttling, Incorporate Throttling
time, What Will Lost Data Cost You?, What Will Downtime Cost You?, Don’t Forget Unix mtime, atime, and ctime
losing through data loss, What Will Lost Data Cost You?
losing through downtime, What Will Downtime Cost You?
Unix, recording for files, Don’t Forget Unix mtime, atime, and ctime
time-to-data, Time-to-Data
toolkit, creating for new servers, Take an Inventory
torture-testing backup utilities, Torture-Testing Backup Programs, Torture-Testing Backup Programs, Torture-Testing Backup Programs
transaction, Transaction
transaction logs, Don’t Go Overboard, Transaction log
database, importance of, Don’t Go Overboard
transfer speed, Transfer Speed
tuple, Row
twos complement platforms, The Little Endian That Couldn’t

U

ufsdump utility, Demystifying dump, Dumpster Diving
filesystem backup, Dumpster Diving
filesystems, inactivity for backups, Demystifying dump
Ultra Density Optical (UDO) recording format, UDO recording format
Unix systems, Demystifying dump, Dumpster Diving
dump utility, Dumpster Diving
filesystem, Demystifying dump
user error causing data loss, Be Ready for Anything: 10 Types of Disasters

user-managed backups, User-Managed Backups, Physical Backups Without rman, Debunking Hot-Backup Myths
utime, using to reset atime, The atime can be reset—with a penalty

V

validated systems, Plan for the Worst
vandalism, losing data from, Be Ready for Anything: 10 Types of Disasters
very critical applications, Aggressive Requirements
very large applications, Aggressive Requirements
very large database (VLDB), Simultaneous Backup of One Client to Many Drives
very large system (VLS), Simultaneous Backup of One Client to Many Drives
virtual tape cartridges (VTCs), Disk-As-Tape: Virtual Tape Cartridges
virtual tape libraries, Disk-As-Tape: Virtual Tape Cartridges (see VTL)
virtual tape library, Disk-As-Tape: Virtual Tape Libraries (see VTL)
VMware, VMware and Miscellanea, Using Bare-Metal Recovery to Migrate to VMware, VMware Architecture, VMware Architecture, VMware Backups, Back up suspended virtual machine files, Using Bare-Metal Recovery to Migrate to VMware
  architecture, VMware Architecture
  backups, VMware Backups
  backups, suspended virtual machine files, Back up suspended virtual machine files
  bare-metal recovery, Using Bare-Metal Recovery to Migrate to VMware
  ESX Server, VMware Architecture
volatile filesystem backups, Volatile Filesystems, Missing or Corrupted Files, Referential Integrity Problems, Corrupted

or Unreadable Backup, Torture-Testing Backup Programs, Using Snapshots to Back Up a Volatile Filesystem
backup utilities, Torture-Testing Backup Programs
missing/corrupted files, Missing or Corrupted Files
referential integrity problems, Referential Integrity Problems
snapshots, Using Snapshots to Back Up a Volatile Filesystem
unreadable backups, Corrupted or Unreadable Backup
volatile filesystems, Volatile Filesystems, Missing or Corrupted Files, Referential Integrity Problems, Corrupted or Unreadable Backup, Using Snapshots to Back Up a Volatile Filesystem
backing up with snapshots, Using Snapshots to Back Up a Volatile Filesystem
corrupted or unreadable backups, Corrupted or Unreadable Backup
missing or corrupted files, Missing or Corrupted Files
referential integrity problems, Referential Integrity Problems
volume groups, backing up all on system, IBM’s mksysb and savevg Utilities
volume managers, configuration information, recording, Take an Inventory
Volume Shadow Copy Service (VSS), Client/Server Architecture Using Nonproprietary Tools
volume verification, Volume Verification
volumes, tapes versus (backups), Why the Word “Volume” Instead of “Tape”?
VTL, Disk drives: Very flexible, Disk-As-Tape: Virtual Tape Libraries, Advantages of VTLs, How do you eject virtual tapes?, How do you eject virtual tapes?
ejecting, How do you eject virtual tapes?
integrated, How do you eject virtual tapes?
versus disk-as-disk targets, Advantages of VTLs
VXA, VXA

W

Walker, John, Bacula
Weichinger, Stefan G., Amanda
Windows, An Overview, ntbackup, Using System Restore in Windows, Backing Up with the dump Utility, BackupPC Features, The BackupPC Community, Bacula, Plug-in Support for File Daemons, Bacula Features
    BackupPC and, BackupPC Features, The BackupPC Community
    Bacula open-source backup, Bacula, Plug-in Support for File Daemons
    dump utility, Backing Up with the dump Utility
    ntbackup and, An Overview, ntbackup
    System Restore and, Using System Restore in Windows
    VSS support, Bacula Features
Windows bare-metal recovery, Linux and Windows, Commercial Solutions, Live or alternate boot?, Image level or filesystem level?, Complete disk or separate partitions?, Alt-Boot Full Image Method, Alt-Boot Partition Image Method, Alt-Boot Filesystem Method, Automate Bare-Metal Recovery with G4L
    alt-boot filesystem method, Alt-Boot Filesystem Method
    alt-boot full image method, Alt-Boot Full Image Method
    alt-boot method, Live or alternate boot?, Image level or filesystem level?, Complete disk or separate partitions?
        complete disk versus separate partitions, Complete disk or separate partitions?
        image versus filesystem level, Image level or filesystem level?
    alt-boot partition image method, Alt-Boot Partition Image Method
    with G4L, Automate Bare-Metal Recovery with G4L
WinZip, tar
WORM recording methods, WORM recording methods

write-once-read-many (WORM), Cost
writes, committed and partially committed (tracking), Be Ready for Anything: 10 Types of Disasters

X

xfsdump utility, incompatiblity with dump (native versions), Different Backup Format

Z

Zmanda enterprise support, Community and Support Options

About the Author

W. Curtis Preston has specialized in designing data protection systems since 1993, and has designed such systems for many environments, both large and small. His lively prose and wry, real-world approach has made him a popular author and speaker.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of *Backup & Recovery* is an Indian gavial (sometimes spelled gharial), a resident of deep, fast-moving rivers in India and neighboring countries. Growing six to seven meters long, the gavial is one of the largest members of the crocodilian family. It is most notable for its extremely long, narrow snout.

This snout, which is lined with razor-sharp teeth, is perfectly suited for catching and eating fish, the gavial's principal food. The narrow shape results in little water resistance, making rapid side-to-side snatches easy. The many sharp teeth are effective for holding onto struggling, slippery fish. The gavial's short, poorly muscled legs makes moving on land very awkward, and thus it only emerges from the water for nesting and basking in the sun. Like other crocodiles, the gavial has often been accused of being a man-eater. Findings of human remains and jewelry in gavial stomach ha s has perpetuated this belief, but since Hindi burial rituals in the gavial's habitat involve setting the cremated body afloat in the river, this is probably where these items come from. However, this animal is as poorly suited for eating humans as it is well-suited for eating fish.

Gavials are highly endangered, and came close to extinction in the 1970s. Thanks to conservation efforts there has been some recovery of the gavial population. They have been protected since the 1970s, but males are still sometimes hunted for their snouts, which are said to

have aphrodisiac properties. Gavials can also become caught in fishing nets, resulting in their death.

In summary, in the words of this book's author: "Let's see . . . huge, intimidating, ugly creature that's not actually harmful to humans . . . That sounds like backups to me!"

The cover image is a 19th-century engraving from the Dover Pictorial Archive. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed.

About the Author

W. Curtis Preston has been specializing in data protection since 1993, when he was responsible for backups at a $35 billion credit card company with 24/7 availability requirements. It was there that Curtis selected and used his first commercial backup and recovery product. He left that company to go into consulting "to get out of backups and become a real system administrator." His first assignment was at the headquarters of another very large corporation that, as it turned out, had no decent backup system. So much for getting out of backups. An Oracle backup script he had written came in so handy there that he decided to publish it in his first article. The hundred or so emails he received from around the world were enough to give Curtis the publishing bug. He suddenly realized that there was quite a demand for the knowledge that he had acquired over the years. He immediately began working on Unix Backup & Recovery, his first book. This was followed by more articles, another book (Using SANs and NAS), and many speaking opportunities around the world. Curtis has advised some of the largest companies in the world on their data protection systems, including three of the Fortune 10.

Curtis is now the vice president of data protection for GlassHouse Technologies, the largest independent provider of storage-related professional services, including complete operational management of your backup system. He is now considered by many people to be the industry's foremost expert on backup and data protection.